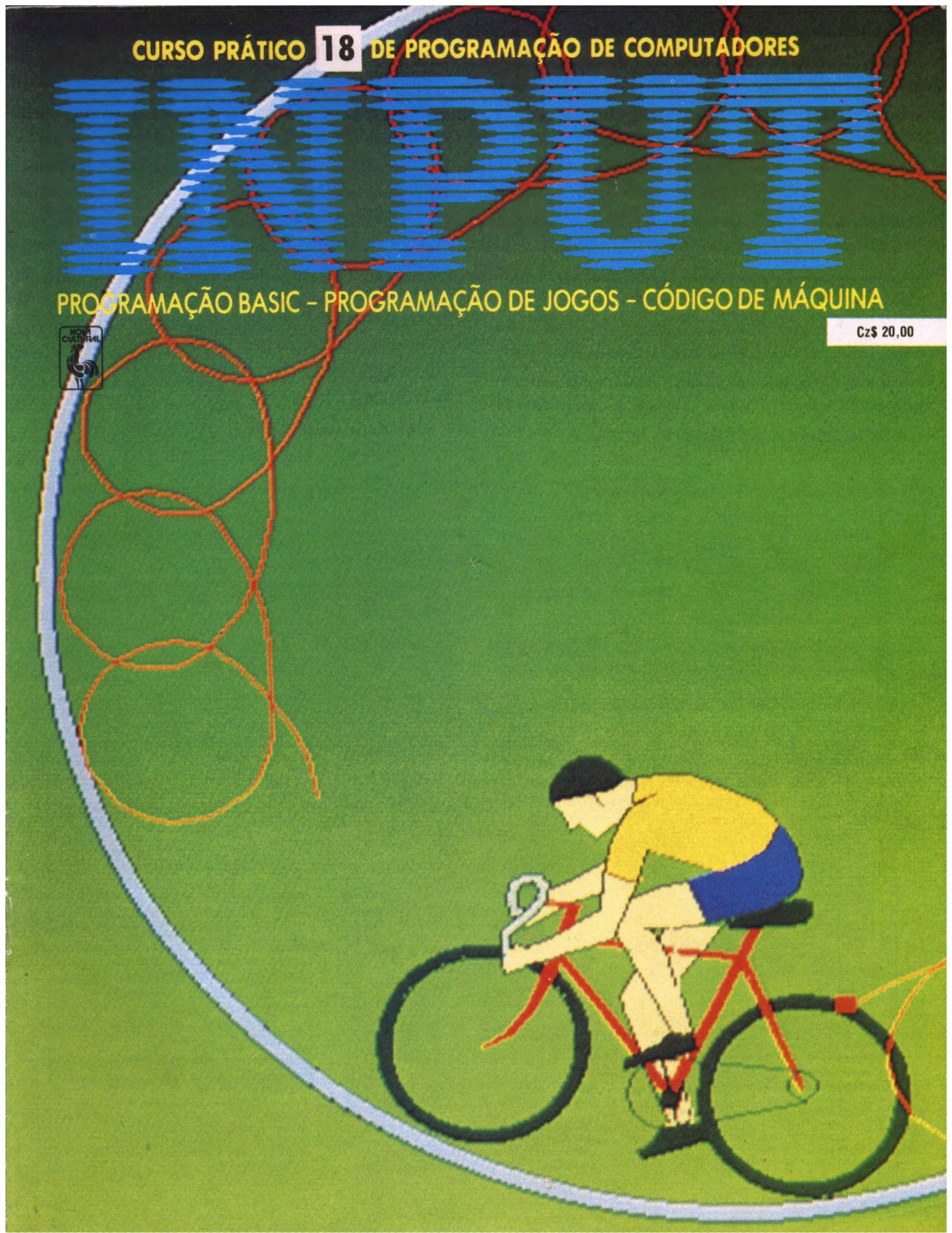
CURSO PRÁTICO 18 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 20,00



INPUT

Vol. 2 Nº 18

NESTE NÚMERO

CÓDIGO DE MÁQUINA

MOVIMENTE FIGURAS NA TELA

Rotinas em código dentro de programas BASIC. Como criar uma figura móvel na memória. O que são caracteres definidos pelo usuário. Movimento. Faça um tanque de guerra que atira e um sapo que pula 341

PROGRAMAÇÃO DE JOGOS

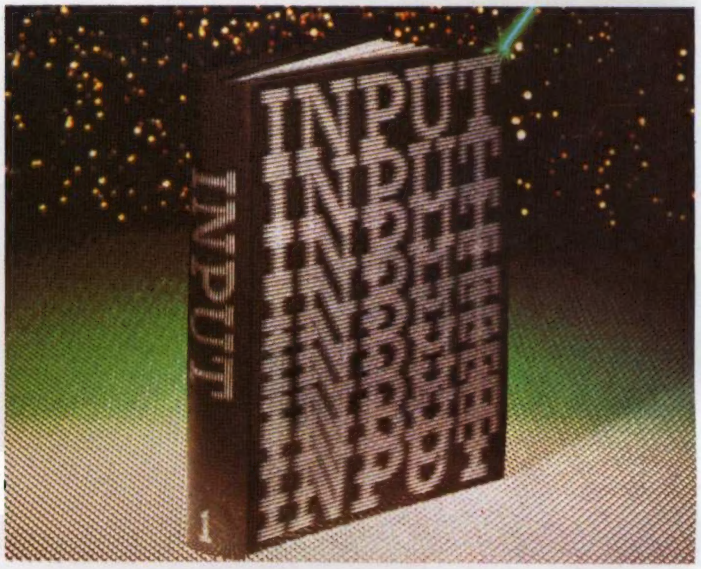
PROGRAMAÇÃO PARA JOYSTICKS

Compatibilidade dos diferentes tipos de joystick. Utilização de joysticks em programas de jogos em BASIC. Como mover sprites com o joystick. Crie uma alça de mira animada. Um teste para seu joystick 348

PROGRAMAÇÃO BASIC

MAIS REQUINTE EM SEUS DESENHOS

Como empregar **SIN** e **COS** na elaboração de desenhos sofisticados. Faça um relógio mecânico. Como posicionar os números. Desenhe os ponteiros. Acrescente um alarme. Como adicionar números romanos. O uso de funções matemáticas na criação de desenhos abstratos 354



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: **1. Pessoalmente** — por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em São Paulo os endereços são: Rua Brigadeiro Tobias, 773 (Centro); Av. Industrial, 117 (Santo André); e, no Rio de Janeiro: Rua da Passagem, 93 (Botafogo). **2. Por carta** — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. **3. Por telex** — Utilize o nº (011) 33670 ABSA. Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.
Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na Agência do Correio. **Atenção**: Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor chefe: Paulo de Almeida
Editor de texto: Cláudio A.V. Cavalcanti
Editor de Arte: Eduardo Barreto
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, José Maria de Oliveira, Grace A. Arruda, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz
Secretário Gráfico: Antonio José Filho

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M.E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda. Campinas, SP.
Tradução: Maria Fernanda Sabbatini
Adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Ferrucio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atilio Roberto Bonon
Preparador de Texto/Coordenador: Eliel Silveira Cunha

Preparadores de Texto: Ana Maria Dilguerian, Antonio Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian, Maria Teresa Galluzzi, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria do Carmo Leme Monteiro, Maria Luiza Simões, Maria Teresa Martins Lopes.

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.
Edição organizada pela Editora Nova Cultural Ltda.
(Artigo 15 da Lei 5 988, de 14/12/1973).

Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Dívisão Gráfica da Editora Abril S.A.

MOVIMENTE FIGURAS NA TELA

- COMO CRIAR UMA FIGURA MÓVEL NA MEMÓRIA
- MOVIMENTOS MAIS RÁPIDOS
- UM SAPO QUE PULA E UM TANQUE QUE ATIRA

Não é só o programador experiente que pode se divertir com linguagem de máquina. Utilize alguns programas pequenos, escritos em código, para dar mais vida a seus programas em BASIC.

Linguagem de máquina é coisa bem complicada. Para a maioria dos usuários de micros, ela não passa de um amontoado de números sem sentido. Porém, se você usar algumas rotinas em código dentro de programas BASIC, logo notará as vantagens da linguagem de máquina e ficará mais motivado a enfrentar as dificuldades.

Os programas que aqui apresentamos utilizam o BASIC para colocar programas em código de máquina na memória do micro. Com isto, pode-se movimentar figuras em alta resolução muito mais rapidamente que usando só BASIC.

Os programas são específicos para cada computador. O MSX e o Apple dispõem de comandos BASIC com velocidade satisfatória, e seus programas não precisam de linguagem de máquina para produzir os mesmos resultados dos outros micros. Já publicamos um programa desse tipo para o ZX-81.

Para montar os gráficos das figuras 1 e 2, precisamos fazer três coisas. Primeiro, criar na memória do Spectrum a "grade" ou quadriculado que conterá o desenho em alta resolução. Esse quadriculado será composto por *caracteres definidos pelo usuário*. Segundo, elaborar um programa que movimente o desenho na tela. Terceiro, trocar os caracteres gráficos no quadriculado pela figura que queremos mover — no nosso caso, o tanque e o sapo.

CARACTERES DEFINIDOS PELO USUÁRIO

O Spectrum permite ao usuário mudar o formato de determinados caracteres (caracteres definidos pelo usuário, ou UDGs). Cada caractere é formado por 64 pontos — alguns com a cor do fundo (INK) e alguns com a cor da frente (PAPER) —, num arranjo de oito por oito. Desde que não ultrapassemos os limites deste quadrado, podemos escrever um programa que dê ao caractere o formato que desejarmos.

Existem 21 caracteres desse tipo: A até U, inclusive. Como mostra a figura 1, é possível juntar os caracteres — conjuntos de oito por oito pontos — para formar figuras maiores. Pode-se ter, por exemplo, duas figuras com três por três caracteres ao mesmo tempo (ainda sobrarão 2 UDGs), ou cinco figuras com dois por dois caracteres.

Tanto o sapo quanto o tanque apresentados neste artigo usam uma figura de três por três caracteres, mais ou menos assim:

A B C

D E F

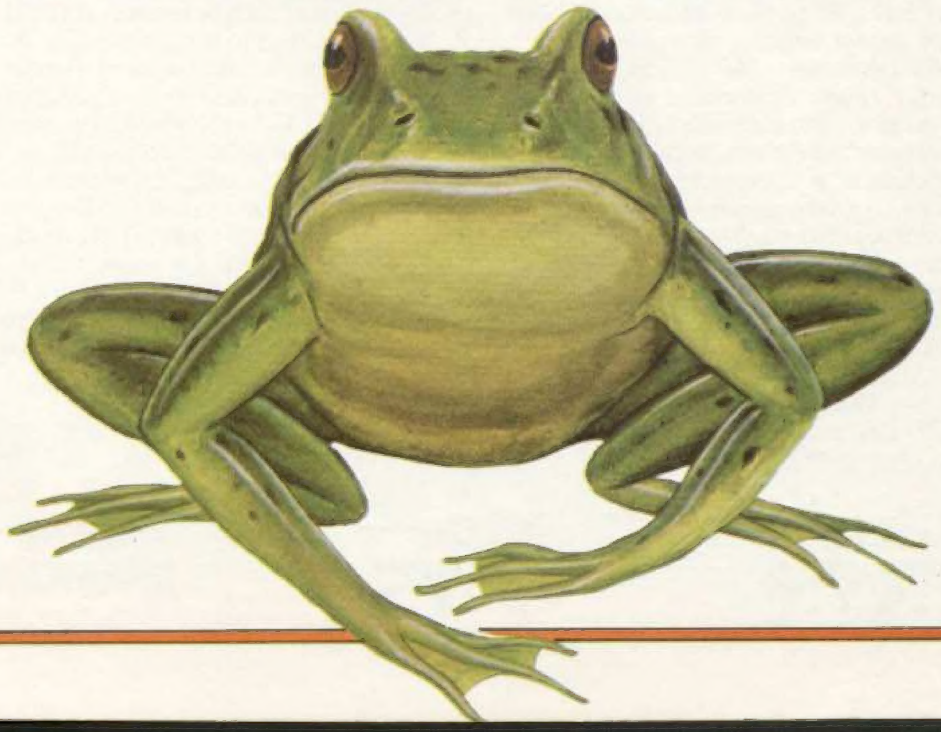
G H I

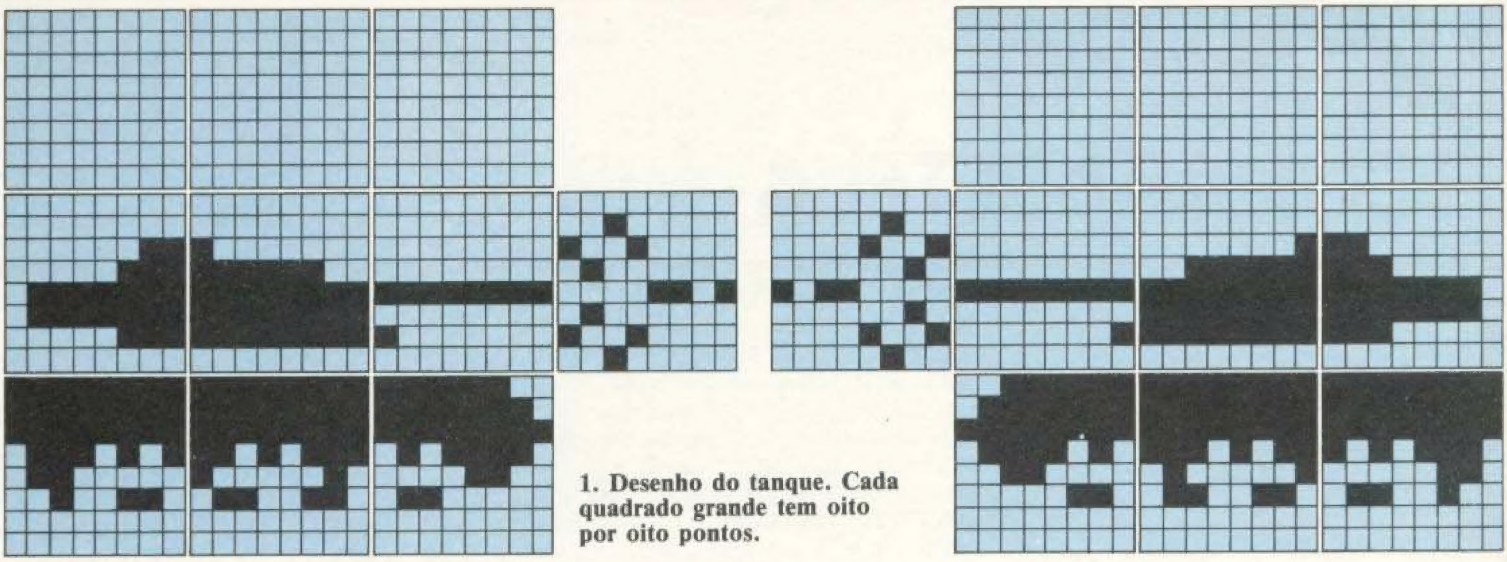
Pode-se colocar isto na tela empregando **PRINT AT**, em BASIC; porém, a melhor alternativa é usar a rotina em linguagem de máquina criada pelo programa a seguir.

```
 10 IF PEEK 23733=127 THEN
CLEAR 32399: LET B=32400:  LET
Z=0
 20 IF PEEK 23733=255 THEN
CLEAR 65199: LET B=65200: LET
Z=1
 30 FOR N=B TO B+129: READ A:
POKE N,A: NEXT N
 40 IF Z=1 THEN  POKE 65258,
178: POKE 65259,254: POKE
65277,179: POKE 65278,254
 50 SAVE "Padrao"CODE B,130
 60 STOP
100 DATA 24,55,1,22,0,0,32,32,
32,22,0,0,32,32,32,22,0,0,32,
32
110 DATA 32,22,0,0,144,145,146
,22,0,0,147,148,149,22,0,0,150
,151,152,22
120 DATA 0,0,153,154,155,22,0,
0,156,157,158,22,0,0,159,160,
161,58,146,126
130 DATA 254,1,1,18,0,40,8,56,
4,203,33,24,2,14,0,221,33,147,
126,221
140 DATA 9,58,137,92,71,62,24,
144,221,119,1,60,221,119,7,60,
221,119,13,58
150 DATA 136,92,71,62,33,144,
221,119,2,221,119,8,221,119,14
,221,229,62,2,205
160 DATA 1,22,209,1,18,0,205,
60,32,201
```

Talvez você ache que esta é uma maneira complicada de fazer o que alguns poucos **PRINT AT** fariam. Mas existem algumas razões para a substituição:

1. Uma vez digitado e gravado, este pro-





1. Desenho do tanque. Cada quadrado grande tem oito por oito pontos.

grama pode ser usado quantas vezes você quiser.

2. Quando chamada de dentro de um programa BASIC, a rotina criada imprimirá os caracteres numa velocidade muito maior que os **PRINT AT**.

Sem saber código de máquina é impossível compreender o que os números de cada linha **DATA** significam, mas tentaremos dar uma idéia de como funciona o programa em BASIC.

As linhas 10 e 20 determinam onde a rotina em código será colocada, dependendo do tamanho da memória de seu Spectrum — 16K ou 48K. Note que os comandos podem não funcionar em micros de 16K com expansão. Se este for o caso, mude a linha 20 para:

```
20 CLEAR 65199: LET B=65200:
LET Z=1
```

Os números incluídos nas linhas **DATA** são transferidos para a memória do computador pela linha 30.

A linha 50 grava o programa. Você pode mudar o nome dele, se quiser.

As linhas 100 a 160 contêm os números que, quando colocados na memória, formam a rotina em código.

Esta rotina é gravada de uma maneira diferente. Após digitar e rodar o programa, o computador pedirá ao usuário que ligue o gravador e aperte qualquer tecla, a fim de gravar a rotina em linguagem de máquina.

Já que a rotina em código está dentro da memória — colocada pelo programa BASIC ou lida por meio do cassete —, vamos usá-la. Pressione **NEW** e **ENTER**, para que o programa antigo não atrapalhe o novo.

```
 20 LET print=32400: LET B=
32402: IF PEEK 23733=255 THEN
LET print=65200: LET B=65202
 90 BORDER 7: PAPER 7: INK 4:
CLS
100 LET Y=8: LET X=15: LET Y1=
8: LET X1=15: LET Z=1
110 LET A$=INKEY$
120 IF A$="z" AND X>0 THEN
LET X1=X-1: LET Z=1
130 IF A$="x" AND X<29 THEN
LET X1=X+1: LET Z=2
140 IF A$="p" AND Y>0 THEN
LET Y1=Y-1
150 IF A$="l" AND Y<18 THEN
LET Y1=Y+1
170 LET X=X1: LET Y=Y1
180 PRINT AT Y,X;: POKE B,Z:
RAND USR print
190 GOTO 110
```

Note que a palavra "print" em minúsculas deve ser digitada letra por letra, ao contrário de **PRINT**, que fica na tecla **P**.

Este programa permite a movimentação de figuras na tela usando as teclas Z, X, P e L. Quando ele for rodado, veremos que foram criadas duas figuras com três por três caracteres; a segunda é formada por JKLMNOPQR. Pode-se, assim, obter duas versões do mesmo gráfico. No caso do tanque, um aponta para a direita, quando vai nesta direção, e outro aponta para a esquerda, quando se movimenta para a esquerda.

Veremos também que, conforme a figura se move, deixa para trás um rastro que não foi planejado. Para corrigir isto, acrescente a linha:

```
160 PRINT AT Y,X;: POKE B,0:
RAND USR print
```

Ela produz uma figura de três por três caracteres em branco, que vai apagando as posições já ocupadas.

UM TANQUE DE GUERRA

Para transformar os caracteres na figura de um tanque, acrescente as próximas linhas. Elas montam o tanque apontando para a direita no quadriculado 1 e para a esquerda no quadriculado 2.

```
  10 FOR N=USR "a" TO USR "r"+7
: READ A: POKE N,A: NEXT N
1000 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0
1010 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,255,0,1,0
1020 DATA 0,0,1,63,255,255,255,
0,0,0,192,224,254,254,224,0
1030 DATA 63,127,255,122,48,6,0
,0,255,255,255,235,65,102,0,0
1040 DATA 255,255,255,174,6,100
,0,0
1050 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0
1060 DATA 0,0,0,0,0,0,0,0,0,0,0,3
,7,127,127,7,0
1070 DATA 0,0,128,252,255,255,2
55,0,0,0,0,0,255,0,128,0
1080 DATA 255,255,255,117,96,38
,0,0,255,255,255,215,130,102,0,
0
1090 DATA 252,254,255,94,12,96,
0,0
```

Resta criar os dois caracteres que irão "armar" o tanque. Aperte **BREAK** e digite:

```
 10 FOR N=USR "a" TO USR "t"+7
: READ A: POKE N,A: NEXT N
115 IF INKEY$=" " THEN  GOTO
200
200 IF Z=2 THEN  GOTO 300
```

```
 210 FOR N=X-1 TO 0 STEP -1
 220 PRINT  INK 5;AT Y+1,N;CHR$
 162
 230 PAUSE 1
 240 PRINT AT Y+1,N;" "
 250 NEXT N
 260 GOTO 110
 300 FOR N=X+3 TO 31
 310 PRINT  INK 5;AT Y+1,N;CHR$
 163
 320 PAUSE 1
 330 PRINT AT Y+1,N;" "
 340 NEXT N
 350 GOTO 110
1100 DATA 0,4,9,2,176,2,9,4,0,3
2,144,64,13,64,144,32
```

Agora a tecla **SPACE** pode ser usada para dar tiros.

UM SAPO

Para transformar as figuras em um sapo, você precisará livrar-se do tanque. Para isso, grave o programa e depois pressione <NEW> e <ENTER>.

O tanque e o programa que o movimentava serão anulados, mas a rotina em código que cuida do quadriculado permanecerá na memória até que o micro seja desligado.

Digite **CLEAR 32399** no Spectrum de 16K, ou **CLEAR 65199**, no de 48K, reservando o topo da memória para a rotina em código. Digite, depois, **LOAD "" CODE** e ligue o gravador para carregar a rotina em código do cassete.

Com a rotina na memória do micro, digite e rode o programa que cria o sapo.

```
  30  RESTORE 5000: FOR N=USR "
a" TO USR "r"+7: READ A: POKE N
,A: NEXT N
5000 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0
5010 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,0,1,1
5020 DATA 0,0,0,0,0,128,192,176
,0,0,0,0,0,0,0,0
5030 DATA 4,15,31,63,127,254,24
8,127,96,240,224,192,64,32,156,
192
5040 DATA 0,0,0,0,0,0,0,0
5050 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,0,1,3,7
5060 DATA 8,28,27,70,255,254,25
2,249,0,0,0,0,0,0,0,0
5070 DATA 7,7,15,30,54,38,70,70
,250,196,C,0,0,0,0,0
5080 DATA 0,0,1,1,3,6,2,0,140,1
44,16,32,32,48,32,0
5090 DATA 0,0,0,0,0,0,0,0
```

Para controlar o movimento do sapo, digite:

```
 10 BORDER 0: PAPER 0: INK 4:
 BRIGHT 1: CLS
 20 LET P=32400: IF PEEK 23733
 =255 THEN  LET P=65200
100 PRINT AT 10,0;: RAND USR P
: IF INKEY$="" THEN  GOTO 100
110 RESTORE 1000: FOR F=1 TO 5
120 READ A,B,C: POKE P+2,A:
PRINT AT B,C;: RAND USR P
130 PAUSE 2: CLS : NEXT F
150 PRINT AT 10,12;: RAND USR
P: IF INKEY$="" THEN  GOTO 150
200 FOR F=1 TO 5
210 READ A,B,C: POKE P+2,A:
PRINT AT B,C;: RAND USR P
220 PAUSE 2: CLS : NEXT F
230 GOTO 100
1000 DATA 1,10,0,2,7,3,2,5,6,2,
7,9,1,10,12,1,10,12,2,7,15,2,5,
18,2,7,21,1,10,24
```

O Apple não precisa de linguagem de máquina para movimentar uma figura em alta resolução: basta criar uma tabela de figuras na memória do micro e usar o comando **DRAW**.

Se não dispusermos de um editor de figuras, o trabalho poderá ser bem complicado. Mas você não será obrigado a usá-lo, pois o programa a seguir cria o tanque e o movimenta.

```
10  HOME :E = 35000: HIMEM: E
20 F =  INT (E / 256): POKE 232
,E - F * 256: POKE 233,F
30  FOR I = E TO E + 41 + 14 *
32
40  READ A: POKE I,A
50  NEXT
60  HGR : SCALE= 1: ROT= 0:X =
130:Y = 90:F = 0: GOTO 410
70 LX = X:LY = Y:LF = F
80  GET K$: IF K$ = "" THEN 80
90  IF K$ = "P" AND Y > 10 THEN
 Y = Y - 2: GOTO 140
100  IF K$ = "L" AND Y < 140 TH
EN Y = Y + 2: GOTO 140
110  IF K$ = "Z" AND X > 12 THE
N X = X - 3:F = 0: GOTO 140
120  IF K$ = "X" AND X < 220 TH
EN X = X + 3:F = 1: GOTO 140
130  IF K$ = " " THEN 490
140  IF LX = X AND LY = Y THEN
80
150  IF LF = 0 THEN 250
160  HCOLOR= 0
170  DRAW 1 AT LX,LY
180  DRAW 2 AT LX,LY + 8
190  DRAW 3 AT LX + 8,LY
200  DRAW 4 AT LX + 8,LY + 8
210  DRAW 5 AT LX + 16,LY
220  DRAW 6 AT LX + 16,LY + 8
230  IF F = 0 THEN 410
240  GOTO 330
250  HCOLOR= 0
260  DRAW 9 AT LX,LY
270  DRAW 10 AT LX,LY + 8
280  DRAW 11 AT LX + 8,LY
290  DRAW 12 AT LX + 8,LY + 8
300  DRAW 13 AT LX + 16,LY
310  DRAW 14 AT LX + 16,LY + 8
320  IF F = 0 THEN 410
330  HCOLOR= 3
340  DRAW 1 AT X,Y
350  DRAW 2 AT X,Y + 8
360  DRAW 3 AT X + 8,Y
370  DRAW 4 AT X + 8,Y + 8
380  DRAW 5 AT X + 16,Y
390  DRAW 6 AT X + 16,Y + 8
400  GOTO 70
410  HCOLOR= 3
420  DRAW 9 AT X,Y
430  DRAW 10 AT X,Y + 8
440  DRAW 11 AT X + 8,Y
450  DRAW 12 AT X + 8,Y + 8
460  DRAW 13 AT X + 16,Y
470  DRAW 14 AT X + 16,Y + 8
480  GOTO 70
490  IF F = 0 THEN 560
500  HCOLOR= 3
510  DRAW 7 AT X + 24,Y
520  FOR I = 1 TO 100: NEXT
530  HCOLOR= 0
540  DRAW 7 AT X + 24,Y
550  GOTO 80
560  HCOLOR= 3
570  DRAW 8 AT X - 8,Y
580  FOR I = 1 TO 100: NEXT
590  HCOLOR= 0
600  DRAW 8 AT X - 8,Y
610  GOTO 80
2000  DATA  20 ,0 ,42 ,0 ,74 ,0
 ,106 ,0 ,138 ,0 ,170 ,0 ,202 ,
0 ,234 ,0 ,10 ,1 ,42 ,1 ,74 ,1
,106 ,1 ,138 ,1 ,170 ,1 ,202 ,1
 ,234 ,1 ,10 ,2 ,42 ,2 ,74 ,2 ,
106 ,2 ,138 ,2
2010  DATA    72 ,73 ,73 ,218 ,
219 ,219 ,74 ,73 ,41 ,213 ,63 ,
223 ,155 ,41 ,45 ,45 ,173 ,59 ,
63 ,63 ,191 ,73 ,9 ,45 ,213 ,21
9 ,219 ,19 ,0 ,0 ,0,0
2020  DATA     40 ,45 ,45 ,45 ,
213 ,63 ,63 ,63 ,55 ,45 ,45 ,45
 ,173 ,251 ,31 ,63 ,87 ,109 ,73
 ,209 ,59 ,223 ,159 ,73 ,73 ,13
7 ,219 ,219 ,155 ,0 ,0 ,0
2030  DATA    72 ,73 ,73 ,218 ,
219 ,219 ,106 ,73 ,73 ,218 ,59
,63 ,63 ,46 ,45 ,45 ,45 ,213 ,6
3 ,63 ,63 ,55 ,45 ,45 ,45 ,173
,219,219 ,155 ,0 ,0, 0
2040  DATA     40 ,45 ,45 ,45 ,
213 ,63 ,63 ,63 ,55 ,45 ,45 ,45
 ,173 ,59 ,255 ,31 ,55 ,77 ,73
,141 ,27 ,255 ,59 ,87 ,73 ,73 ,
209 ,219 ,219 ,19 ,0 ,0
2050  DATA     72 ,73 ,73 ,218
,219 ,219 ,74 ,73 ,73 ,218 ,219
 ,219 ,42 ,45 ,45 ,45 ,213 ,219
 ,219 ,19 ,77 ,73 ,137 ,219 ,21
9 ,155 ,0 ,0 ,0 ,0 ,0 ,0
2060  DATA     40 ,45 ,45 ,77 ,
218 ,63 ,63 ,63 ,46 ,45 ,45 ,45
 ,213 ,59 ,63 ,31 ,87 ,73 ,109
,209 ,219 ,27 ,191 ,73 ,73 ,137
```

```
 ,219 ,219 ,155 ,0 ,0 ,0
2070  DATA      72 ,73 ,73 ,218
,219 ,251 ,106 ,105 ,73 ,218 ,2
19 ,27 ,87 ,73 ,109 ,213 ,219 ,
219 ,23 ,77 ,77 ,137 ,219 ,219
,159 ,0 ,0 ,0 ,0 ,0 ,0 ,0
2080  DATA      72 ,73 ,73 ,218
,251 ,219 ,74 ,9 ,77 ,213 ,251
,219 ,19 ,13 ,109 ,73 ,218 ,223
 ,219 ,74 ,9 ,77 ,213 ,27 ,223
,155 ,0 ,0 ,0 ,0 ,0 ,0
2090  DATA      72 ,73 ,73 ,218
,219 ,219 ,74 ,73 ,73 ,218 ,219
 ,219 ,42 ,45 ,45 ,45 ,213 ,219
 ,219 ,83 ,73 ,73 ,213 ,219 ,21
9 ,19 ,0 ,0 ,0 ,0 ,0 ,0
2100  DATA      72 ,45 ,45 ,173
,59 ,63 ,63 ,191 ,45 ,45 ,45 ,1
73 ,27 ,31 ,63 ,191 ,9 ,109 ,73
 ,218 ,255 ,219 ,74 ,73 ,73 ,21
8 ,219 ,219 ,2 ,0 ,0, 0
2110  DATA      72 ,73 ,73 ,218
,219 ,219 ,74 ,73 ,9 ,213 ,63 ,
63 ,255 ,42 ,45 ,45 ,45 ,213 ,6
3 ,63 ,63 ,55 ,45 ,45 ,45 ,173
,219 ,219 ,155 ,0 ,0 ,0
2120  DATA      40 ,45 ,45 ,45 ,
213 ,63 ,63 ,63 ,55 ,45 ,45 ,45
 ,173 ,59 ,31 ,31 ,63 ,14 ,77 ,
73 ,213 ,59 ,223 ,191 ,73 ,73 ,
137 ,219 ,219 ,155 ,0, 0
2130  DATA      72 ,73 ,73 ,218
,219 ,219 ,42 ,77 ,73 ,209 ,219
 ,27 ,63 ,46 ,45 ,45 ,109 ,218
,63 ,63 ,63 ,46 ,109 ,73 ,209 ,
219 ,219 ,19 ,0 ,0 ,0 ,0
2140  DATA      40 ,45 ,45 ,45 ,
213 ,63 ,63 ,63 ,55 ,45 ,45 ,45
 ,173 ,27 ,63 ,31 ,31 ,78 ,73 ,
109 ,218 ,251 ,59 ,87 ,73 ,73 ,
209 ,219 ,219 ,19 ,0 ,0
```

A linha 10 reserva o topo da memória para a tabela de figuras. O Apple é então informado da localização da mesma pela linha 20. O **FOR...NEXT** das linhas 30 a 50 monta a tabela na memória.

A linha 60 liga a tela de alta resolução, estabelece a escala e a orientação do desenho. Além disso, coloca as coordenadas do centro da tela em **X** e **Y**. Em seguida, o programa é desviado para a linha 140, para que uma imagem inicial do tanque seja feita.

As linhas 70 a 140 movimentam o tanque através das teclas Z, X, P e L. Tratam também de não deixá-lo ultrapassar os limites da tela. As variáveis **F** e **LF** são sinalizadores que indicam a direção atual e antiga do tanque; desenham e apagam a figura na orientação correta. A linha 140 evita que o tanque seja apagado e desenhado, sem sair do lugar. A linha 130 verifica se a tecla de espaço foi pressionada, saltando para a linha 420, onde começa a porção do programa que dispara um tiro — na direção correta!

As linhas 150 a 320 apagam o tanque de sua última posição, conforme o valor de **LF**. As linhas 330 a 480 desenham-no na nova posição, conforme o valor de F.

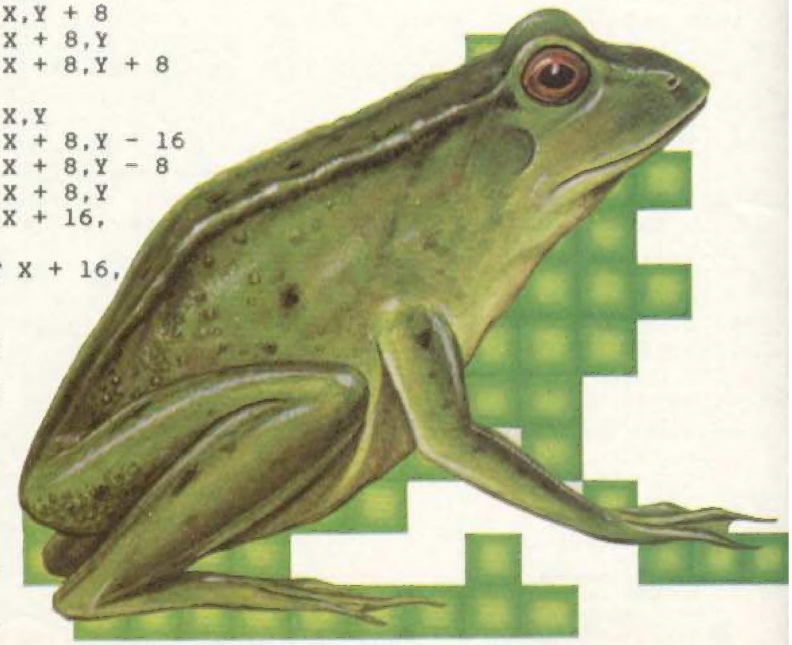
Note que as linhas **DATA** 2000 a 2140 foram geradas com o programa editor apresentado no artigo da página 316. Os blocos do tanque são numerados de cima para baixo e, depois, da esquerda para a direita.

HCOLOR = 3 desenha o tanque em branco. Se utilizarmos **HCOLOR = 1**, teremos a cor verde, mas o desenho será ligeiramente deformado. Isto ocorre devido à maneira como o Apple usa as cores em alta resolução.

UM SAPO

O programa a seguir monta uma tabela correspondente ao sapo da figura 2.

```
10  HOME :E = 35000: HIMEM: E
20  POKE 233, INT (E / 256): PO
KE 232,E - 256 *  PEEK (233)
30  FOR I = E TO E + 41 + 10 *
32: READ A: POKE I,A: NEXT I
40  HGR : SCALE= 1: ROT= 0
100  GOSUB 1800: GOSUB 1300
110  GOSUB 1000
120  GET K$: IF K$ <  > " " THE
N 120
130  GOSUB 1200: GOSUB 1000
140  GOSUB 1500: GOSUB 1300
150  GOSUB 1040: GOSUB 1400
160  GOSUB 1200: GOSUB 1500
170  GOSUB 1040: GOSUB 1300
180  GOSUB 1600: GOSUB 1040
190  GOSUB 1400: GOSUB 1200
200  GOSUB 1600: GOSUB 1040
210  GOSUB 1300: GOSUB 1700
220  GOSUB 1000: GOSUB 1400
230  GOSUB 1200: GOSUB 1700
240  GOSUB 1000: GOTO 100
1000  DRAW 1 AT X,Y
1010  DRAW 2 AT X,Y + 8
1020  DRAW 3 AT X + 8,Y
1030  DRAW 4 AT X + 8,Y + 8
1035  RETURN
1040  DRAW 5 AT X,Y
1050  DRAW 6 AT X + 8,Y - 16
1060  DRAW 7 AT X + 8,Y - 8
1070  DRAW 8 AT X + 8,Y
1080  DRAW 9 AT X + 16,
Y - 16
1090  DRAW 10 AT X + 16,
Y - 8
1100  RETURN
1200  HCOLOR= 0:
RETURN
1300  HCOLOR= 3:
RETURN
1400  FOR I = 1
TO 200: NEXT I
1410  RETURN
1500 X = 90:Y =
50: RETURN
1600 X = 160:Y =
50: RETURN
1700 X = 230:Y =
90: RETURN
1800 X = 30:Y = 90: RETURN
2000  DATA   20 ,0 ,42 ,0 ,74 ,
0 ,106 ,0 ,138 ,0 ,170 ,0 ,202
 ,0 ,234 ,0 ,10 ,1 ,42 ,1 ,74 ,1
 ,106 .1 ',138 ,1 ,170 ,1 ,202 ,
1 ,234 ,1 ,10 ,2 ,42 ,2 ,74 ,2
,106 ,2 ,138 ,2
2010  DATA    0 ,72 ,73 ,73 ,21
8 ,219 ,219 ,74 ,73 ,73 ,218 ,2
19 ,219 ,74 ,73 ,73 ,218 ,219 ,
219 ,74 ,73 ,9 ,213 ,223 ,219 ,
19 ,0 ,0 ,0 ,0 ,0 ,0
2020  DATA    0 ,72 ,73 ,77 ,26
 ,63 ,255 ,155 ,73 ,45 ,45 ,213
 ,63 ,63 ,255 ,10 ,45 ,45 ,45 ,
213 ,59 ,63 ,63 ,55 ,45 ,45 ,77
 ,209 ,63 ,63 ,63 ,23
2030  DATA    0 ,72 ,73 ,73 ,21
8 ,219 ,219 ,74 ,73 ,73 ,218 ,2
19 ,219 ,74 ,73 ,73 ,218 ,219 ,
219 ,46 ,77 ,73 ,209 ,219 ,59 ,
31 ,6 ,0 ,0 ,0 ,0 ,0
2040  DATA    0 ,8 ,109 ,73 ,20
9 ,219 ,59 ,63 ,46 ,109 ,73 ,20
9 ,219 ,219 ,119 ,77 ,73 ,209 ,
219 ,27 ,159 ,73 ,45 ,77 ,218 ,
219 ,219 ,2 ,0 ,0 ,0 ,0
2050  DATA    0 ,72 ,73 ,73 ,21
8 ,219 ,219 ,74 ,73 ,73 ,26 ,22
3 ,219 ,83 ,73 ,9 ,173 ,27 ,255
 ,219 ,74 ,73 ,105 ,218 ,219 ,2
19 ,2 ,0 ,0 ,0 ,0 ,0
2060  DATA    0 ,72 ,73 ,73 ,21
8 ,219 ,219 ,74 ,73 ,73 ,218 ,2
19 ,219 ,74 ,73 ,73 ,26 ,223 ,2
19 ,83 ,73 ,9 ,173 ,59 ,255 ,21
9 ,2 ,0 ,0 ,0 ,0 ,0
2070  DATA    0 ,72 ,73 ,45 ,21
3 ,63 ,223 ,155 ,73 ,41 ,45 ,21
3 ,59 ,63 ,223 ,74 ,109 ,109 ,2
18 ,255 ,251 ,10 ,77 ,41 ,141 ,
27 ,255 ,27 ,23 ,0 ,0 ,0
2080  DATA    0 ,104 ,9 ,109 ,2
09 ,219 ,251 ,51 ,77 ,77 ,137 ,
219 ,219 ,159 ,9 ,77 ,73 ,218 ,
219 ,255 ,74 ,77 ,73 ,218 ,219
```



```
,219 ,2 ,0 ,0 ,0 ,0 ,0
2090  DATA    0 ,72 ,9 ,77 ,209
 ,27 ,63 ,223 ,74 ,41 ,13 ,173
,27 ,255 ,27 ,23 ,45 ,45 ,45 ,1
73 ,27 ,63 ,63 ,63 ,46 ,45 ,45
,77 ,26 ,223 ,63 ,63
2100  DATA    0 ,40 ,45 ,109 ,1
41 ,219 ,223 ,63 ,78 ,73 ,73 ,2
18 ,219 ,219 ,74 ,73 ,73 ,218 ,
219 ,219 ,74 ,73 ,73 ,218 ,219
,219 ,2 ,0 ,0 ,0 ,0 ,0
```

Use a barra de espaço para fazer o sapo pular.

Para definir e mover os desenhos das figuras 1 e 2, precisamos fazer três coisas. Primeiro, criar na memória do computador uma "grade" ou quadriculado onde possamos desenhar o que quisermos. Este quadriculado será composto por *caracteres definidos pelo usuário*. Segundo, transferir o padrão do desenho desejado — representado por números em linha **DATA** — para o quadriculado. Terceiro, elaborar um programa em BASIC que desenhe e movimente o tanque e o sapo.

Um caractere definido pelo usuário (UDG) é um "quadro", dentro do qual se coloca parte de um desenho em alta resolução. O quadro é composto por 64 pontos dispostos num arranjo de oito por oito pontos. Cada ponto pode ser preto ou branco em **PMODE 4,1** (este é o **PMODE** mais flexível; em outros é preciso definir dois ou quatro pontos de cada vez).

Se colocarmos vários desses caracteres lado a lado, poderemos construir desenhos maiores e mais detalhados. À diferença de outros computadores, o TRS-Color não possui UDGs embutidos. Mas porções de sua memória se comportarão como tal, se fizermos um programa para isto.

Teoricamente, dispomos de espaço para criar muitos caracteres, mas seria trabalhoso usar vários deles. Na prática, quatro ou cinco bastam.

COMO CRIAR O QUADRICULADO

Para produzir o tanque da figura 1, ou o sapo da figura 2, precisamos primeiro fazer com que uma porção da memória se comporte como UDGs. Os dois desenhos requerem um quadriculado de três por três caracteres — 24 por 24 pontos.

Se usássemos o BASIC para desenhar uma figura destas, precisaríamos de um **PSET** para desenhar cada ponto, ou seja, 576 **PSET** ao todo. Assim, um programa em linguagem de máquina será mais rápido.

Digite:

```
10 CLEAR 200,32000
20 FOR I=32000 TO 32110
30 READ N
40 POKE I,N
50 NEXT
60 CLS
90 PRINT "APERTE  QUAL
QUER   TECLA  PARA
GRAVAR O PROGRAMA EM
LINGUAGEM  DE MAQUI
NA"
100 B$=INKEY$
11' IF B$="" THE
N 100
120 CSAVEM "P
RPADRAO",
32000,321
10,32000
```

```
130 DATA 190,127,188,134,3,183,
125,111,183,125,112,134,8,183,1
25,113
140 DATA 182,125,250,39,50,206,
126,44,74,198,72,61,51,203,166,
192
150 DATA 167,132,48,136,32,122,
125,113,38,244,134,8,183,125,11
3,48
160 DATA 137,255,1,122,125,111,
38,230,134,3,183,125,111,48,137
,0
170 DATA 253,122,125,112,38,216
,57,95,231,132,48,136,32,122,12
5,113
180 DATA 38,246,134,8,183,125,1
13,48,137,255,1,122,125,111,38,
232
190 DATA 134,3,183,125,111,48,1
37,0,253,122,125,112,38,218,57
```

Este programa cria um outro, em linguagem de máquina. Sua gravação deve ser feita pelo próprio programa; portanto, tenha o gravador por perto — os programas aqui apresentados não funcionam com o drive de disquetes conectado.

Além de produzir desenhos em alta resolução numa velocidade muito maior, o programa em código ocupa bem menos memória que uma rotina BASIC.

Se você conhece a linguagem de máquina do 6809, não entenderá o que os números nas linhas **DATA** significam, mas pode ter uma idéia de como funciona o programa em BASIC.

A linha 10 reserva a memória necessária para acomodar o programa em código. As linhas 90 a 120 transferem o programa da memória para a fita.

UM TANQUE DE GUERRA

Para desenhar o tanque são necessárias duas figuras de três por três caracteres — uma para o tanque apontando para a direita e outra para o tanque que aponta para a esquerda.

Digite **NEW** para anular o programa antigo — o programa em código permanecerá na memória. Este novo programa cria o tanque mas, por enquanto, não o desenha.

```
10 CLEAR 200,32000
20 FOR I=32300 TO 32443
30 READ N
40 POKE I,N
50 NEXT
60 DATA 0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0
70 DATA 0,0,3,7,127,127,7,0,0,0
,128,252,255,255,255,0
80 DATA 0,0,0,0,255,0,128,0,255
,255,255,117,96,38,0,0
90 DATA 255,255,255,215,130,102
,0,0,252,254,255,94,12,96,0,0
100 DATA 0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0
110 DATA 0,0,0,0,255,0,1,0,0,0,
```

```
1,63,255,255,255,0
120 DATA 0,0,192,224,254,254,22
4,0,63,127,255,122,48,6,0,0
130 DATA 255,255,255,235,65,102
,0,0,255,255,255,174,6,100,0,0
```

Para desenhar e mover o tanque, acrescente as próximas linhas ao programa anterior:

```
5 PCLEAR 5
170 PMODE 4,1
180 PCLS
290 PCLS
300 SCREEN 1,1
310 T=1
320 TP=3500
330 POKE 32700,INT(TP/256)
340 POKE 32701,TP-256*INT(TP/25
6)
350 POKE 32250,T
360 EXEC 32000
370 LP=TP
380 IF PEEK(338)=251 THEN TP=TP
-32:GOTO 440
390 IF PEEK(342)=253 THEN TP=TP
+32:GOTO 440
400 IF PEEK(340)=247 THEN TP=TP
-1:T=2:GOTO 440
410 IF PEEK(338)=247 THEN TP=TP
+1:T=1:GOTO 440
430 GOTO 380
440 IF TP<1536 OR TP>6941 THEN
TP=LP
470 GOTO 330
```

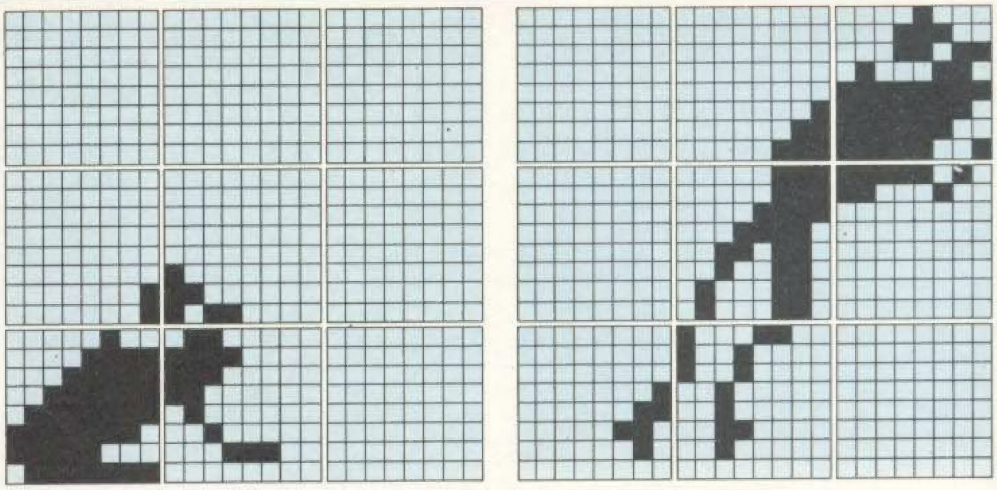
Agora digite **RUN**; usando as teclas P, L, Z e X, você poderá mover o tanque. Este deixará um rastro atrás de si. Para evitar o problema, digite:

```
450 POKE 32250,0
460 EXEC 32000
```

Estas linhas usam o programa em código para imprimir uma figura composta por caracteres em branco.

Para que o canhão atire, acrescente mais algumas linhas ao programa. Você poderá fazê-lo por meio da barra de espaço.

```
160 DIM A(2),B(2),C(2)
190 FOR I=1536 TO 1760 STEP 32
190 FOR I=1536 TO 1760 STEP 32
200 READ N
210 POKE I,N
220 NEXT
230 GET (0,0)-(7,7),A
240 FOR I=1536 TO 1760 STEP 32
250 READ N
260 POKE I,N:NEXT
280 GET (0,0)-(7,7),B
420 IF PEEK(345)=247 GOSUB 500
480 DATA 0,32,144,64,13,64,144,
32
490 DATA 0,4,9,2,176,2,9,4
500 IF T=2 THEN 560
510 YP=INT((TP-1536)/32)+8
520 XP=8*(TP-1533-(YP-8)*32)
530 IF XP>255 THEN 620
540 PUT (XP,YP)-(XP+7,YP+7),A
550 GOTO 600
```



2. Desenho do sapo. A figura pula quando se pressiona a tecla de espaço.

```
560 YP=INT((TP-1536)/32)+8
570 XP=8*(TP-1537-(YP-8)*32)
580 IF XP<0 THEN 620
590 PUT (XP,YP)-(XP+7,YP+7),B
600 FOR I=1 TO 100:NEXT
610 PUT (XP,YP)-(XP+7,YP+7),C
620 RETURN
```

UM SAPO

A figura 2 mostra o desenho de um sapo. Depois que ele for transferido para o quadriculado da memória e puder ser controlado pelo programa em BASIC, sairá pulando tela afora.

Para transferir o desenho do sapo para o quadriculado, utilize o programa a seguir. Antes, porém, use **NEW** para limpar a memória ocupada pelo BASIC.

```
10 CLEAR 200,32000
20 FOR I=32300 TO 32443
30 READ N
40 POKE I,N:NEXT
60 DATA 0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0
70 DATA 0,0,0,0,0,0,1,1,0,0,0,0
,0,128,192,176
80 DATA 0,0,0,0,0,0,0,0,4,15,31
,63,127,254,248,127
90 DATA 96,240,224,192,64,32,15
6,192,0,0,0,0,0,0,0,0
100 DATA 0,0,0,0,0,0,0,0,0,0,0,
0,0,1,3,7,8,28,27,70,255,254,25
2,249
110 DATA 0,0,0,0,0,0,0,0,7,7,15
,30,62,54,70,70
120 DATA 250,196,0,0,0,0,0,0,0,
0,1,1,3,6,2,0
130 DATA 140,144,16,32,32,48,32
,0,0,0,0,0,0,0,0,0
```

Para fazer o sapo pular, digite **NEW** e, em seguida, este programa:

```
10 PCLEAR 5
20 PMODE 4,1
30 PCLS
40 SCREEN 1,1
50 FP=4487
60 POKE 32700,17
70 POKE 32701,135
80 POKE 32250,1
90 EXEC 32000
100 IF PEEK(345)<>247 THEN 100
110 RESTORE
120 FOR I=1 TO 5
130 READ F,FD
140 POKE 32700,INT(FP/256)
150 POKE 32701,FP-256*INT(FP/25
6)
160 POKE 32250,F
170 EXEC 32000
180 FOR J=1 TO 70:NEXT
190 POKE 32250,0
200 EXEC 32000
210 FP=FP+FD
220 NEXT
230 GOTO 80
240 DATA 1,-287,2,-253,2,258,2,
291,1,0
```

Os números da linha **DATA** no final do programa fazem o sapo descrever um arço quando pressionamos a barra de espaço.

MSX

O BASIC do MSX dispensa o uso de linguagem de máquina, pois dispõe de comandos gráficos poderosos e é suficientemente rápido para criar e movimentar figuras em alta resolução.

O programa a seguir cria e movimenta o tanque da figura 1.

```
5 CLEAR 300:COLOR 12,15,12
10 SCREEN 1,3:KEY OFF
20 FOR I=1 TO 16*8
30 READ A:A$=A$+CHR$(A)
40 NEXT I
50 SPRITE$(0)=LEFT$(A$,32)
60 SPRITE$(1)=MID$(A$,33,32)
70 SPRITE$(2)=MID$(A$,65,32)
80 SPRITE$(3)=MID$(A$,97,32)
90 SPRITE$(4)=MID$(A$,33,16)
100 SPRITE$(5)=STRING$(16,0)+MI
D$(A$,81,16)
110 X=100:Y=90:F=2
111 PUT SPRITE 0,(X-16,Y),12,5
112 PUT SPRITE 1,(X+16,Y),12,3
```

```
120 K$=INKEY$:IF K$="" THEN 120
130 IF ASC(K$)=29 AND X>0 THEN
X=X-1:F=2:GOTO 200
140 IF ASC(K$)=28 AND X<210 THE
N X=X+1:F=1:GOTO 200
150 IF ASC(K$)=30 AND Y>0 THEN
Y=Y-1:GOTO 200
160 IF ASC(K$)=31 AND Y<160 THE
N Y=Y+1:GOTO 200
200 ON F GOSUB 500,600
210 GOTO 120
500 PUT SPRITE 0,(X,Y),12,0
510 PUT SPRITE 1,(X+32,Y),12,4
520 RETURN
600 PUT SPRITE 0,(X-16,Y),12,5
610 PUT SPRITE 1,(X+16,Y),12,3
620 RETURN
5000 DATA  0 , 0 , 3 , 7 , 127
, 127 , 7 , 0 , 255 , 255 , 255
 , 117 , 96 , 38 , 0 , 0 , 0 ,
0 , 128 , 252 , 255 , 255 , 255
 , 0 , 255 , 255 , 255 , 215 ,
130 , 102 , 0 , 0
5010 DATA  0 , 0 , 0 , 0 , 255
, 0 , 128 , 0 , 252 , 254 , 255
 , 94 , 12 , 96 , 0 , 0 , 0 , 3
2 , 144 , 64 , 13 , 64 , 144 ,
32 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0
5020 DATA  0 , 4 , 9 , 2 , 176
, 2 , 9 , 4 , 0 , 0 , 0 , 0 , 0
 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
255 , 0 , 1 , 0 , 63 , 127 , 25
5 , 122 , 48 , 6 , 0 , 0
5030 DATA  0 , 0 , 1 , 63 , 255
 , 255 , 255 , 0 , 255 , 255 ,
255 , 235 , 65 , 102 , 0 , 0 ,
0 , 0 , 192 , 224 , 254 , 254 ,
 224 , 0 , 255 , 255 , 255 , 17
4 , 6 , 100 , 0 , 0
```

A linha 5 estabelece as cores da tela — **COLOR** — e reserva espaço na memória para acomodar o cordão **A$**, que é muito longo — **CLEAR 300**.

A linha 10 seleciona a tela de 32 colunas, com sprites grandes e em baixa resolução (32 por 32 pontos). Além disso, desativa a impressão das teclas de função na parte inferior da tela.

O **FOR...NEXT** das linhas 20 a 40 lê as linhas **DATA** do final do programa, onde está definido o padrão do tanque. Estes valores são transferidos para o cordão **A$** (veja página 188).

As linhas 50 a 100 criam os sprites que montam o tanque. Observe que o cordão **A$** é "recortado" pelas funções **MID$** e **LEFT$**. Os sprites criados são: 0, parte posterior do tanque da direita da figura 2; 1, parte dianteira atirando; 2, parte dianteira do tanque da esquerda atirando; 3, parte traseira. Os sprites 4 e 5 contêm a parte dianteira do tanque, cada uma apontada para um lado, sem o tiro.

As linhas 110 a 112 desenham o tanque em sua posição inicial. As linhas 120 a 160 mudam o valor de **X** e **Y**, conforme a tecla do cursor pressionada, movimentando o tanque. As condições que se seguem à conjunção **AND** evitam que o tanque ultrapasse os limites da tela. **F** é um sinalizador da direção em que o tanque aponta.

A linha 200 utiliza **F** para decidir em que direção desenhar o tanque. O desenho propriamente dito é feito pelas subrotinas 500 e 600.

Note que os sprites da parte dianteira e traseira são desenhados em níveis de prioridade diferentes. Se ambos estivessem no mesmo nível, o segundo a ser desenhado faria o primeiro desaparecer. Esta mesma propriedade responde pelo desaparecimento do tanque de sua antiga posição, quando a nova é desenhada.

As próximas linhas farão o tanque atirar ao toque da barra de espaço.

```
170 IF ASC(K$)=32 THEN IF F=1 T
HEN F=3 ELSE F=4
200 ON F GOSUB 500,600,700,800
700 PUT SPRITE 0,(X,Y),12,0
710 PUT SPRITE 1,(X+32,Y),12,1
720 FOR I=1 TO 50:NEXT
730 F=F-2:GOSUB 500:RETURN
800 PUT SPRITE 0,(X-16,Y),12,2
810 PUT SPRITE 1,(X+16,Y),12,3
820 FOR I=1 TO 50:NEXT
830 F=F-2:GOSUB 600:RETURN
```

Para produzir o tiro, o programa usa o sinalizador **F**. Conforme se tenha pressionado a barra de espaço, também são utilizados os sprites 1 e 2, correspondentes à parte dianteira dando tiro — nos dois sentidos.

Se você quiser sprites grandes em alta resolução, faça as seguintes modificações:

```
10 SCREEN 1,2:KEY OFF
111 PUT SPRITE 0,(X-8,Y),12,5
112 PUT SPRITE 1,(X+8,Y),12,3
500 PUT SPRITE 0,(X,Y),12,0
510 PUT SPRITE 1,(X+16,Y),12,4
600 PUT SPRITE 0,(X-8,Y),12,5
610 PUT SPRITE 1,(X+8,Y),12,3
700 PUT SPRITE 0,(X,Y),12,0
710 PUT SPRITE 1,(X+16,Y),12,1
800 PUT SPRITE 0,(X-8,Y),12,2
810 PUT SPRITE 1,(X+8,Y),12,3
```

UM SAPO

O programa a seguir cria o sapo da figura 2 e o faz pular ao toque da barra de espaço.

```
5 CLEAR 300:COLOR 12,15,12
10 SCREEN 1,3:KEY OFF
20 FOR I=1 TO 12*8
30 READ A:A$=A$+CHR$(A)
40 NEXT I
50 SPRITE$(0)=LEFT$(A$,32)
60 SPRITE$(1)=MID$(A$,33,32)
70 SPRITE$(2)=MID$(A$,65,32)
110 X=50:Y=120
111 PUT SPRITE 0,(X,Y),12,0
120 K$=INKEY$:IF K$="" THEN 120
170 IF ASC(K$)=32 THEN GOSUB 50
0
210 GOTO 120
500 PUT SPRITE 0,(X+32,Y-64),12
,1
510 PUT SPRITE 1,(X+16,Y-32),12
,2
520 FOR I=1 TO 100:NEXT
530 PUT SPRITE 0,(X+64,Y-64),12
,1
540 PUT SPRITE 1,(X+48,Y-32),12
,2
550 FOR I=1 TO 100:NEXT
560 PUT SPRITE 0,(X+96,Y),12,0
570 PUT SPRITE 1,(X+48,209),12,
2
580 X=X+96
590 RETURN
5000 DATA  0 , 0 , 0 , 0 , 0 ,
0 , 1 , 1 , 4 , 15 , 31 , 63 ,
127 , 254 , 248 , 127 , 0 , 0 ,
 0 , 0 , 0 , 128 , 192 , 176 ,
96 , 240 , 224 , 192 , 192 , 32
 , 28 , 0
5010 DATA  0 , 0 , 0 , 0 , 0 ,
1 , 3 , 7 , 7 , 7 , 15 , 30 , 5
4 , 38 , 70 , 70 , 8 , 28 , 27
, 70 , 255 , 254 , 252 , 249 ,
250 , 196 , 0 , 0 , 0 , 0 , 0 ,
 0
5020 DATA  0 , 0 , 0 , 1 , 3 ,
6 , 2 , 0 , 0 , 0 , 0 , 0 , 0 ,
 0 , 0 , 0 , 140 , 144 , 144 ,
32 , 32 , 48 , 32 , 0 , 0 , 0 ,
 0 , 0 , 0 , 0 , 0 , 0
```

A preparação do computador e a criação dos sprites a partir das linhas **DATA** são muito parecidas com o programa do tanque, só que o laço **FOR...NEXT** é mais curto.

A diferença é que não precisamos mover o sapo, apenas fazê-lo saltar quando a tecla de espaço for pressionada. As linhas 120 e 170 detectam a ocorrência de pressão na tecla.

A sub-rotina 500 é responsável pela trajetória que o sapo descreve. Para entender como os sprites foram montados, use **PUT SPRITE** no modo imediato, após ter parado o programa com **<CNTRL> <STOP>**.

Para obter o mesmo efeito com sprites de alta resolução, faça as seguintes modificações:

```
10 SCREEN 1,2:KEY OFF
500 PUT SPRITE 0,(X+16,Y-32),12
,1
510 PUT SPRITE 1,(X+8,Y-16),12,
2
530 PUT SPRITE 0,(X+32,Y-32),12
,1
540 PUT SPRITE 1,(X+24,Y-16),12
,2
560 PUT SPRITE 0,(X+48,Y),12,0
570 PUT SPRITE 1,(X+48,209),12,
2
580 X=X+48
```

PROGRAMAÇÃO PARA JOYSTICKS

Os joysticks tornam seus jogos muito mais profissionais. E não se preocupe: você não precisa saber linguagem de máquina para utilizá-lo — o BASIC é suficiente.

Os jogos vendidos no comércio e os produzidos em casa apresentam, em geral, uma diferença evidente: os primeiros oferecem ao usuário a opção de usar um joystick; os segundos, não. Todavia, não é necessário mergulhar nas profundezas da programação em código para incluir este periférico nos programas de jogos.

Você verá neste artigo como utilizar um joystick em programas BASIC, tornando seus jogos mais profissionais e, sobretudo, muito mais divertidos.

Antes de fazer um programa que o inclua, é preciso obter um joystick compatível com o micro em questão. Os programas que aqui apresentamos devem ser utilizados com o joystick padrão disponível para cada computador. Observações a respeito se encontram antes de cada programa. Mais informações sobre joysticks em geral podem ser obtidas no artigo da página 287.

S

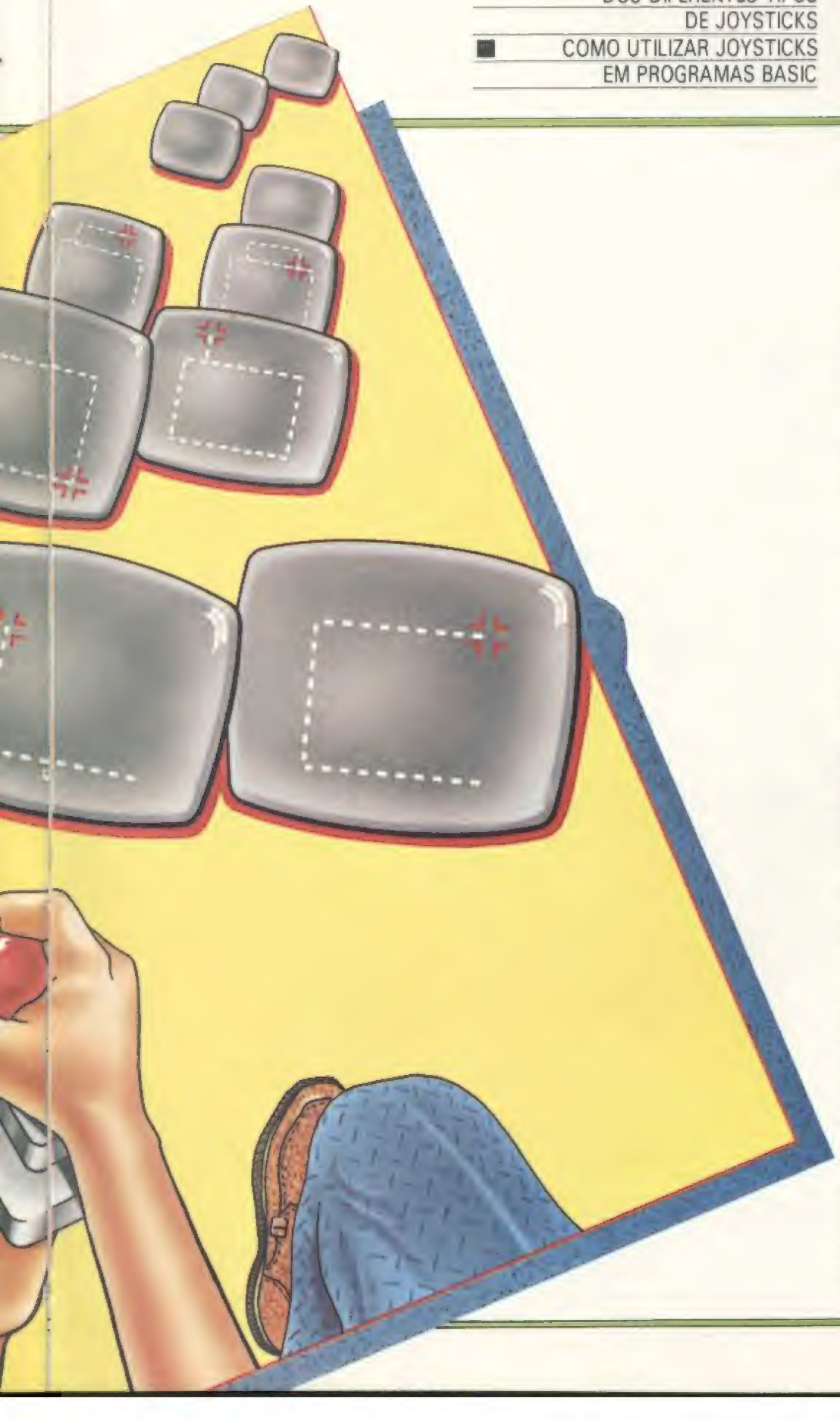
Existem vários tipos de joystick compatíveis com o Spectrum. Não é possível escrever um programa que funcione com todos eles, pois cada um tem sua própria maneira de se comunicar com a máquina. Assim, optamos por um programa que funcione com o tipo mais comum de joystick: Atari.

UMA ALÇA DE MIRA ANIMADA

A primeira parte do programa permite a movimentação de uma alça de mira na tela do Spectrum.

```
100 BORDER 1: PAPER 1: INK 7:
OVER 1: CLS : INK 8
110 FOR n=USR "a" TO USR "h"+7
: READ a: POKE n,a: NEXT n
130 LET s=0: LET x=15: LET y=
10
140 PRINT  OVER 1;AT y,x;CHR$
148;CHR$ 149;AT y+1,x;CHR$ 150
;CHR$ 151
200 GOSUB 500
480 GOTO 200
500 LET i$=INKEY$: IF i$=""
THEN  RETURN
505 LET i=CODE i$
510 PRINT  OVER 1;AT y,x;CHR$
```

- COMPATIBILIDADE DOS DIFERENTES TIPOS DE JOYSTICKS
- COMO UTILIZAR JOYSTICKS EM PROGRAMAS BASIC
- UMA ALÇA DE MIRA ANIMADA
- TORNE SEUS JOGOS MAIS PROFISSIONAIS
- UM TESTE PARA SEU JOYSTICK



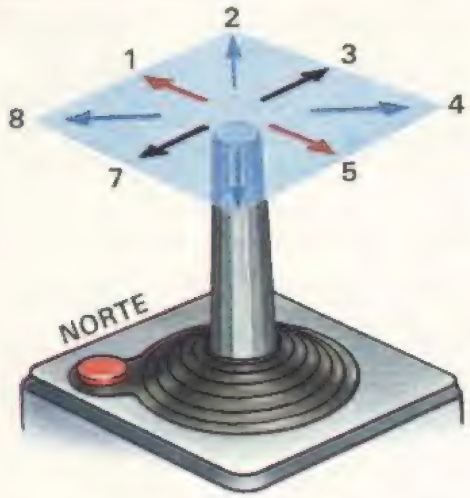
```
148;CHR$ 149;AT y+1,x;CHR$ 150
;CHR$ 151
520 IF i=57 AND y>1 THEN  LET
y=y-1
530 IF i=56 AND y<20 THEN  LET
y=y+1
540 IF i=55 AND x<30 THEN  LET
x=x+1
550 IF i=54 AND x>0 THEN  LET
x=x-1
560 PRINT  OVER 1;AT y,x;CHR$
148;CHR$ 149;AT y+1,x;CHR$ 150
;CHR$ 151
570 RETURN
1000 DATA 14,27,127,31,15,7,15,
31
1010 DATA 0,0,0,0,0,192,112,188
1020 DATA 31,29,30,15,3,1,1,3
1030 DATA 206,30,124,248,224,64
,64,224
1040 DATA 12,12,12,12,252,252,0
,0
1050 DATA 48,48,48,48,63,63,0,0
1060 DATA 0,0,252,252,12,12,12,
12
1070 DATA 0,0,63,63,48,48,48,48
```

O programa começa estabelecendo as cores do vídeo. Em seguida, utilizando os valores das linhas **DATA** 1000 a 1070, cria oito caracteres definidos pelo usuário. O programa não usa todos os caracteres, pois eles desenham não só a alça de mira, mas também um pato. Este será utilizado pela parte restante do programa, que apresentaremos num próximo artigo.

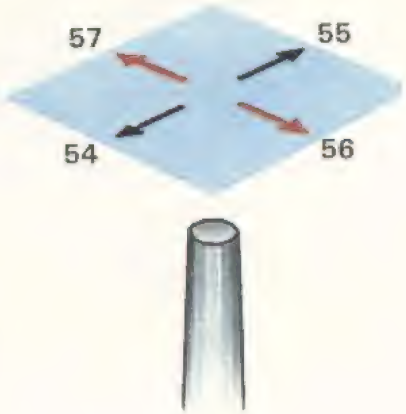
Criados os caracteres, a linha 130 acerta a posição inicial da mira na tela. Além disso, zera o contador de pontos — ou escore —, que também aguarda a próxima parte do programa para ser utilizado.

A linha 140 usa o comando **PRINT** para colocar na tela a metade superior da mira, usando dois caracteres definidos pelo usuário. Depois, coloca a metade inferior, usando dois outros caracteres. A alça de mira é controlada pela sub-rotina que começa na linha 500, chamada linha 200.

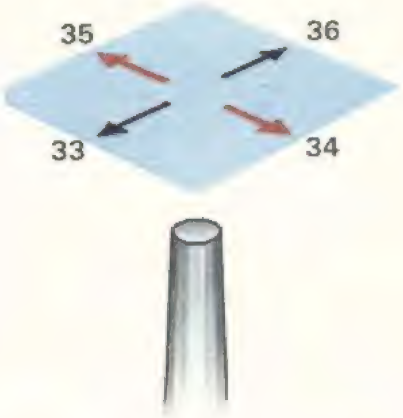
A sub-rotina que controla o joystick usa a função **INKEY$** para saber em que direção o bastão está virado. Isto é possível porque o joystick envia caracteres ao micro, como se fizesse parte do teclado. Os códigos desses caracteres estão na figura 2. As quatro direções principais dão os valores 57, 56, 55 e 54.



1. A função *STICK(n)* do MSX assume valores diferentes conforme a direção dada pelo bastão do joystick.



2. O joystick do Atari envia caracteres ao Spectrum. Os códigos mudam conforme a direção.



3. Estes são os códigos dos caracteres enviados ao computador pelo joystick do TK-85.

A primeira linha da sub-rotina — linha 500 — verifica se o joystick está na posição central. Nenhum caractere será enviado se o bastão ocupar esta posição. Em seguida, a linha 505 guarda o código do caractere na variável **i**, usando a função **CODE**.

A linha 510 apaga a mira, pois é a segunda vez que **PRINT OVER 1** foi usado (a primeira foi na linha 140). **OVER 1** faz com que o gráfico impresso na primeira vez desapareça quando é usado pela segunda vez.

Agora que a posição antiga foi apagada, a nova pode ser calculada. Ela vai depender da direção em que o bastão do joystick for empurrado pelo jogador. A linha 520 percebe o movimento para cima; a linha 530 percebe os movimentos para baixo; e as linhas 540 e 550 percebem os movimentos laterais.

A sub-rotina termina colocando a alça de mira em sua nova posição. Observe que, como é a primeira vez que a figura é desenhada nesta posição, ela aparece normalmente.

Para permitir um movimento contínuo da mira, a linha 480 traz um **GOTO 200**, fazendo com que a sub-rotina do joystick seja chamada repetidamente.

SPECTRUM

Como o ZX-81 não permite a movimentação de figuras usando só o BASIC, a mira desenhada pelo programa a seguir é apenas um ponto gráfico (um quarto de caractere).

O programa deve ser utilizado com o joystick do TK-85 da Microdigital.

```
 20 LET Y=20
 30 GOTO 170
 40 LET LX=X
 50 LET LY=Y
100 LET A$=INKEY$
110 IF A$="" THEN GOTO 100
115 LET A=CODE A$
120 IF A=33 AND X>0 THEN LET X=
X-1
130 IF A=36 AND X<63 THEN LET X
=X+1
140 IF A=34 AND Y>0 THEN LET Y=
Y-1
150 IF A=35 AND Y<40 THEN LET Y
=Y+1
160 UNPLOT LX,LY
170 PLOT X,Y
180 GOTO 40
```

As linhas 10 e 20 armazenam a posição central da tela gráfica em **X** e **Y**. A linha 30 desvia o programa para a linha 170, para que o primeiro ponto seja impresso. Se ela for suprimida, o primeiro ponto só aparecerá após algum movimento do joystick.

As linhas 40 e 50 guardam os últimos valores assumidos por **X** e **Y** em **LX** e **LY**, para que o programa saiba onde apagar a última posição do ponto.

As linhas de 100 a 150 controlam o joystick. A linha 100 promove uma "varredura" do teclado. Isto é necessário porque o joystick envia caracteres ao micro, como se fizesse parte do teclado. A linha 110 repete a linha 100 até que algum movimento seja feito no joystick. Ao ocorrer o movimento, a linha 115 guarda o código do caractere enviado na variável **A**, usando **CODE**.

Os códigos dos caracteres enviados pelo joystick são 35 (para cima), 34 (para baixo), 33 (esquerda) e 36 (direita) (veja a figura 3). As linhas 150, 140, 120 e 130 detectam, respectivamente, estes movimentos. As condições sobre **X** e **Y**, que se seguem à conjunção **AND** nestas mesmas linhas, evitam que o ponto ultrapasse os limites da tela.

Finalmente, a linha 160 apaga o ponto de sua última posição, usando **UNPLOT**, e a linha 170 coloca-o em sua nova posição com **PLOT**. A linha 180 volta ao princípio.

MSX

O MSX tem uma função especial — **STICK (N)** —, que facilita muito o controle de joysticks por programas em BASIC. Além disso, essa função permite que nosso programa seja utilizado mesmo por quem não possui um joystick.

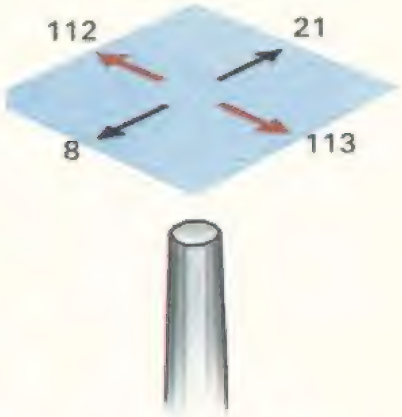
COMO MOVER SPRITES COM O JOYSTICK

O programa a seguir foi feito para funcionar com o joystick do HOTBIT conectado à tomada frontal direita.

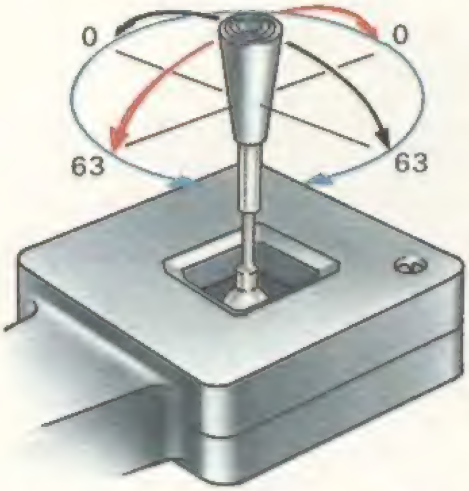
```
10 SCREEN 1,2:KEY OFF
20 FOR I=1 TO 32
30 READ A:A$=A$+CHR$(A)
40 NEXT I
50 SPRITE$(0)=A$
60 PUT SPRITE 0,(115,85),15
170 X=115:Y=85
200 GOSUB 1000
210 GOTO 200
1000 A=STICK(1)
1010 ON A GOTO 1030,1040,1050,1
060,1070,1080,1090,1100
1020 RETURN
1030 Y=Y-1:GOTO 1110
1040 X=X+1:Y=Y-1:GOTO 1110
1050 X=X+1:GOTO 1110
1060 X=X+1:Y=Y+1:GOTO 1110
1070 Y=Y+1:GOTO 1110
1080 X=X-1:Y=Y+1:GOTO 1110
```



```
10 LET X=32
```



4. Cada direção corresponde a um caractere no joystick do TK-2000. Estes são seus códigos.



5. O joystick do TRS-Color é analógico; cada um dos seus dois potenciômetros fornece um valor entre 0 e 63.

6. A leitura do joystick analógico do Apple só pode ser feita por um programa em linguagem de máquina.

```
1090 X=X-1:GOTO 1110
1100 X=X-1:Y=Y-1:GOTO 1110
1110 PUT SPRITE 0,(X,Y),15
1120 RETURN
5000 DATA  3 , 12 , 16 , 32 , 3
2 , 64 , 64 , 127 , 64 , 64 , 3
2 , 32 , 16 , 12 , 3 , 0 , 224
, 152 , 132 , 130 , 130 , 129 ,
 129 , 255 , 129 , 129 , 130 ,
130 , 132 , 152 , 224 , 0
```

A linha 10 ativa a tela de 32 colunas, com sprites grandes em alta resolução — **SCREEN 1,2**. Além disso, impede que o conteúdo das teclas de função seja impresso na parte inferior da tela, usando **KEY OFF**.

As linhas 20 a 50 criam um sprite grande — dezesseis por dezesseis — para a alça de mira. Os números que definem o padrão do sprite são lidos na linha 5000 (veja página 188). A mira é então desenhada em sua posição inicial pela linha 60. A linha 170 guarda esta posição em **X** e **Y**.

A linha 200 chama a sub-rotina de controle do joystick, que movimentará o sprite uma vez. A linha 210 faz com que o processo se repita, possibilitando o movimento contínuo.

As linhas 1000 a 1120 controlam o joystick, por meio da função especial **STICK (1)**. Esta assume diferentes valores, conforme a posição do joystick número 1 (veja a figura 1). **STICK (2)** é usada com o joystick 2; **STICK (0)** possibilita a substituição do joystick pelas teclas do cursor.

A linha 1000 armazena o valor correspondente à posição do joystick em **A**. A linha 1010 usa **ON A GOTO** para desviar o programa conforme o valor de **A** (veja página 76). Isto pode ser feito porque os números que o joystick envia são: 0, em repouso; 1, norte; 2, nordeste; 3, leste; 4, sudeste; 5, sul; 6, sudoeste; 7, oeste e 8, noroeste. As linhas para onde o programa é desviado calculam os novos valores de **X** e **Y** — linhas 1030 a 1100. Quando **A** é igual a zero, o programa vai para a linha 1020.

Depois que a nova posição foi calculada, a linha 1110 desenha o sprite nela. A linha 1120 provoca o retorno da sub-rotina.

Por meio do comando **PDL** (*paddle* — em inglês, raquete) controla-se adequadamente o joystick do Apple. Este micro possui uma entrada de joystick que, na verdade, é uma tomada, onde quase todo tipo de aparelho analógico ou digital pode ser conectado sem maiores dificuldades.

O programa a seguir foi feito totalmente em BASIC; por isso, não controla muito bem o joystick analógico.

```
10  HOME :E = 35000: HIMEM: E
20 F =  INT (E / 256): POKE 232
,E - F * 256: POKE 233,F
30  FOR I = E TO E + 41 + 5 * 3
2
40  READ A: POKE I,A
50  NEXT
60  HGR : SCALE= 1: ROT= 0:X =
130:Y = 90
200  GOSUB 500
210  GOTO 200
500 LX = X:LY = Y
510 A =  PDL (1):B =  PDL (0)
530  IF A = 0 AND Y > 8 THEN Y
= Y - 8: GOTO 580
540  IF A = 255 AND Y < 144 THE
N Y = Y + 8: GOTO 580
550  IF B = 0 AND X > 8 THEN X
= X - 8: GOTO 580
560  IF B = 255 AND X < 266 THE
N X = X + 8: GOTO 580
580  HCOLOR= 0
590  DRAW 5 AT LX,LY
600  HCOLOR= 3
610  DRAW 5 AT X,Y
```

MICRO DICAS

DESCUBRA OS CÓDIGOS

Se seu joystick é de outra marca, experimente este pequeno programa para descobrir quais são os códigos dos caracteres enviados por ele.

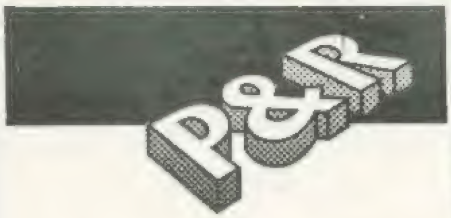
Caso você tenha um Apple, não utilize nenhum programa. Pode ser que seu joystick utilize outro byte analógico. Existem quatro desses bytes, endereços −16284 a −16281. Tente todas as combinações possíveis nas linhas 510 e 520.

```
10 LET A$ = INKEY$
20 IF A$ = "" THEN GOTO 10
30 PRINT CODE A$
40 GOTO 10
```

Conecte o joystick na tomada direita do HOTBIT. Se não funcionar no seu MSX, troque a tomada ou substitua o número 1 por 2 na função **STICK**:

```
10 PRINTSTICK(1);:GOTO 10
```

```
10 GET A$
20 PRINT ASC(A$)
30 GOTO 10
```



Podemos usar joysticks nos outros programas de jogos já publicados em INPUT?

A parte principal do programa fornecido neste artigo pode ser adicionada à maioria dos programas que usam **GET$** ou **INKEY$** para "ler" o teclado. Assim, você poderá aperfeiçoar seus jogos, pois é muito mais conveniente e divertido usar joysticks do que teclado.

As linhas importantes destes programas são: para o Spectrum, linhas 520 a 550; para o ZX81, linhas 100 a 150; para o MSX, linhas 1000 a 1120; para o TK-2000 e Apple, linhas 500 a 530; e para o TRS-Color, as linhas 1000 a 1070, chamadas como sub-rotina (alguns valores terão que ser modificados, conforme o tamanho da figura).

```
620  RETURN
2000  DATA  20 ,0 ,42 ,0 ,74 ,0
 ,106 ,0 ,138 ,0 ,170 ,0 ,202 ,
0 ,234 ,0 ,10 ,1 ,42 ,1 ,74 ,1
,106 ,1 ,138 ,1 ,170 ,1 ,202 ,1
 ,234 ,1 ,10 ,2 ,42 ,2 ,74 ,2 ,
106 ,2 ,138 ,2
2010  DATA  0 ,72 ,9 ,45 ,141
,59 ,31 ,255 ,83 ,45 ,45 ,45 ,2
13 ,63 ,63 ,223 ,74 ,9 ,45 ,173
 ,59 ,255 ,219 ,74 ,9 ,45 ,173
,59 ,63 ,255 ,19 ,0
2020  DATA  0 ,72 ,41 ,45 ,173
 ,251 ,63 ,223 ,74 ,41 ,45 ,141
 ,59 ,63 ,223 ,83 ,73 ,73 ,213
,219 ,219 ,83 ,73 ,73 ,209 ,255
 ,219 ,19 ,0 ,0 ,0 ,0
2030  DATA  0 ,72 ,73 ,73 ,218
 ,219 ,219 ,74 ,73 ,73 ,218 ,21
9 ,219 ,74 ,73 ,73 ,218 ,219 ,2
7 ,119 ,45 ,77 ,137 ,219 ,63 ,2
55 ,6 ,0 ,0 ,0 ,0 ,0
2040  DATA  0 ,40 ,77 ,45 ,141
 ,27 ,63 ,255 ,83 ,45 ,45 ,77 ,
218 ,27 ,63 ,63 ,46 ,109 ,73 ,2
09 ,219 ,27 ,31 ,110 ,77 ,73 ,2
18 ,219 ,63 ,55 ,0 ,0
2050  DATA  0 ,72 ,73 ,73 ,218
 ,27 ,223 ,83 ,73 ,77 ,209 ,219
 ,223 ,83 ,45 ,45 ,45 ,213 ,219
 ,223 ,83 ,73 ,77 ,209 ,219 ,22
3 ,19 ,0 ,0 ,0 ,0 ,0
```

As linhas 10 a 50 criam uma tabela de figuras a partir das linhas **DATA** calculadas com o editor da página 316.

A linha 60 estabelece as condições iniciais de posição, cor e escala na tela gráfica.

A sub-rotina 500 controla o joystick. A linha 210 chama repetidamente a sub-rotina, possibilitando um movimento contínuo.

As linhas 510 e 520 lêem o status dos dois *paddles* conectados ao joystick, por meio dos comandos **PDL(0)** e **PDL(1)**.

As linhas 530 a 560 deslocam a mira continuamente para a esquerda e para cima, a menos que um movimento para baixo ou para a direita seja executado. Isto é o máximo que se pode fazer utilizando apenas o BASIC.

As condições que se seguem às conjunções **AND** evitam que a mira ultrapasse os limites da tela.

As linhas 580 e 590 apagam a mira de sua última posição; as linhas 600 e 610 refazem o desenho na nova.

A tabela de figuras criada a partir das linhas **DATA** inclui um pato, além da mira. Ele será utilizado num próximo artigo.

O programa anterior não funcionará no TK-2000. Faça as modificações indicadas a seguir e use o joystick da Microdigital ou as setas do teclado.

```
510  GET A$
520 A =  ASC (A$)
530  IF A = 112 AND Y > 8 THEN
Y = Y - 8: GOTO 570
540  IF A = 113 AND Y < 144 THE
N Y = Y + 8: GOTO 570
550  IF A = 8 AND X > 8 THEN X
= X - 8: GOTO 570
560  IF A = 21 AND X < 266 THEN
 X = X + 8: GOTO 570
570  REM
```

O joystick envia caracteres ao micro, como se fizesse parte de seu teclado: assim, a linha 510 utiliza **GET$ A$** para obter a nova posição do joystick. A linha 120 guarda o código do caractere enviado em **A**.

Os códigos correspondentes aos movimentos do joystick estão na figura 4. Esses códigos são os mesmos das setas do teclado, motivo pelo qual não é necessário dispor de um joystick para usar o programa.

A linha 530 detecta o movimento para cima; a 540, para baixo; a 550 para a esquerda e a 560 para a direita. As condições que se seguem à conjunção **AND** nestas linhas evitam que a mira ultrapasse os limites da tela.

As operações após **THEN** calculam as novas coordenadas.

Note que o joystick do TK-2000 não é auto-repetitivo. A tecla **REPEAT** contorna parcialmente o problema.

O TRS-Colòr tem uma função em BASIC para controlar o joystick — **JOYSTK** — que facilita muito emprego deste periférico. O computador admite a utilização de dois joysticks, mas o programa abaixo só usa um.

Antes de passar ao programa, conecte seu joystick na tomada traseira marcada com **JOY-DIR**. O programa não funcionará se o drive de disquetes estiver conectado.

UMA ALÇA DE MIRA ANIMADA

Digite a primeira seção do programa e depois rode-a. Você verá uma alça de mira surgir na tela.

```
10 PMODE 3,1
20 FOR K=1536 TO 1868 STEP 32
30 FOR J=0 TO 2
40 READ A:POKE K+J,A
50 NEXT J,K
160 SCREEN 1,0
170 GOTO 170
4000 DATA 252,15,192,192,0,192,
48,3,0,12,12,0,3,48,0
4010 DATA 0,0,0,3,48,0,12,12,0,
48,3,0,192,0,192,251,15,192
```

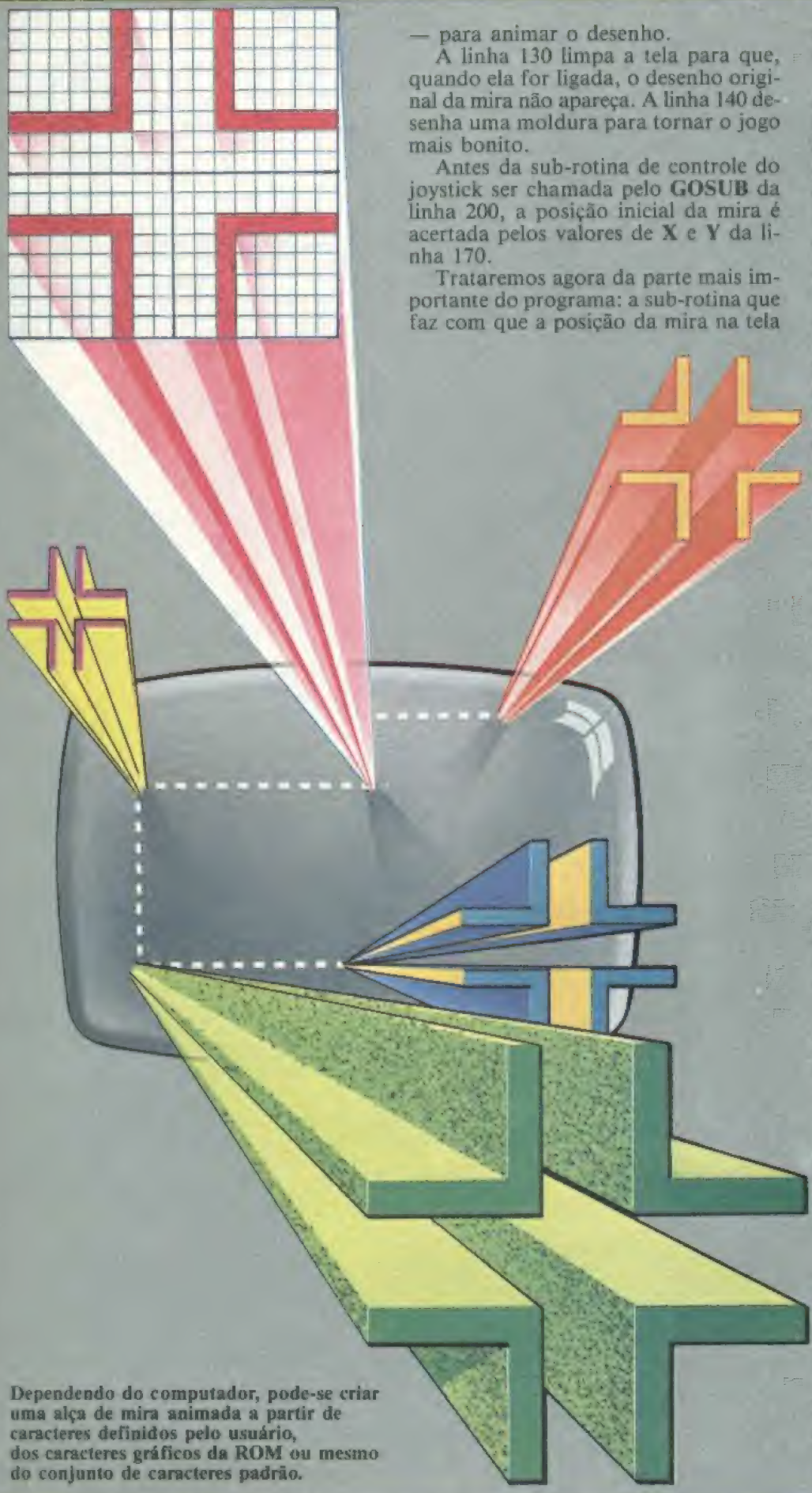
Esta parte do programa é bem simples. As linhas 20 e 50 usam **POKE** para colocar a mira na tela, com os valores dados pelas linhas **DATA** 4000 e 4010.

A linha 160 ativa a tela de alta resolução. A linha 170 é temporária e serve para manter a tela gráfica ligada.

Depois de digitar a segunda parte do programa, você poderá movimentar a mira pela tela usando o joystick.

```
60 DIM S(5),B(5),D(4),H(4)
70 GET (0,0)-(17,11),S,G
130 PCLS
140 LINE(0,0)-(255,191),PSET,B
170 X=127:Y=95
200 GOSUB 1000
210 GOTO 200
1000 J0=JOYSTK(0):J1=JOYSTK(1)
1010 IF J0>58 THEN J0=58
1020 IF J1>59 THEN J1=59
1030 IF X=J0*4+10 AND Y=J1*3+6
THEN 1070
1040 PUT(X-8,Y-5)-(X+9,Y+5),B,P
SET
1050 X=J0*4+10:Y=J1*3+6
1060 PUT (X-8,Y-5)-(X+9,Y+5),S,
OR
1070 RETURN
```

Embora o programa empregue apenas duas matrizes para armazenar desenhos, a linha 60 dimensiona quatro delas — duas para uso futuro. A linha 70, com **GET**, armazena o desenho da mira na matriz **S**. Esta matriz é usada juntamente com a matriz **B** — em branco



Dependendo do computador, pode-se criar uma alça de mira animada a partir de caracteres definidos pelo usuário, dos caracteres gráficos da ROM ou mesmo do conjunto de caracteres padrão.

— para animar o desenho.

A linha 130 limpa a tela para que, quando ela for ligada, o desenho original da mira não apareça. A linha 140 desenha uma moldura para tornar o jogo mais bonito.

Antes da sub-rotina de controle do joystick ser chamada pelo **GOSUB** da linha 200, a posição inicial da mira é acertada pelos valores de **X** e **Y** da linha 170.

Trataremos agora da parte mais importante do programa: a sub-rotina que faz com que a posição da mira na tela corresponda à posição do joystick. Ela vai da linha 1000 à linha 1070.

A linha 1000 usa a função **JOYSTK. JOYSTK (0)** verifica a posição horizontal do **joystick** direito, enquanto **JOYSTK (1)** verifica a posição vertical do mesmo joystick. **JOYSTK (2)** e **JOYSTK (3)** fazem o mesmo para o joystick esquerdo.

Cada uma destas quatro funções pode assumir um valor que vai de 0 a 63, de acordo com a posição dos joysticks.

Na sub-rotina, a linha 1000 usa as variáveis **J1** e **J0** para armazenar o valor de **JOYSTK (1)** e **JOYSTK (0)**: assim, não é preciso escrever o nome todo da função ao longo do programa. Este procedimento é bem comum. Em INPUT, utilizamos **K$** para armazenar o valor de **INKEY$**, por exemplo.

As linhas 1010 a 1020 impedem que a mira saia fora dos limites da tela, levando em conta o tamanho do desenho.

As linhas 1040 a 1060 são responsáveis pela animação. Primeiro, apagam a mira da sua última posição; depois, calculam a nova posição e nela situam a mira, usando **PUT**.

A linha 1040 apaga o desenho em sua última posição colocando a matriz em branco **B**, com **PUT**. A nova posição é calculada a partir de **J1** e **J0** (veja a linha 1050). **X** e **Y** são as novas coordenadas do centro da mira. Observe que, enquantó **J0** é multiplicado por 4, **J1** é multiplicado por 3. Esses fatores de multiplicação são determinados pela resolução da tela que se utilizou. No nosso caso, ela é de 0 a 255 pontos na horizontal, e de 0 a 190 na vertical. Quando multiplicamos os valores dos **JOYSTK**, estamos adaptando o intervalo de 0 a 63 do joystick aos intervalos de 0 a 255 e 0 a 191 da tela. Esses valores são os mesmos para qualquer **PMODE**. Você só precisará alterá-los se quiser considerar o tamanho de outra figura que estiver utilizando. O uso de fatores menores deixa mais espaço para movimentar figuras grandes.

Calculada a nova posição da mira, a linha 1060 coloca o desenho na tela, usando **PUT...OR** para não prejudicar o que estiver desenhado naquelas posições.

A sub-rotina funcionará como já foi explicado após a adição do **RETURN** da linha 1070. Se o joystick não mudasse de posições, a mira ficaria piscando na tela, uma vez que estaria sendo constantemente apagada e desenhada de novo. Para evitar que isso ocorra, a linha 1030 verifica se a posição mudou. Caso o joystick não tenha se movido, o programa continua na linha 1070, evitando as linhas de animação.

MAIS REQUINTE EM SEUS DESENHOS

Já aprendemos a usar funções matemáticas para fazer gráficos circulares. Veremos agora como empregar **SIN** e **COS** na elaboração de desenhos mais sofisticados.

Em artigo anterior, na página 334, vimos algumas das funções matemáticas existentes nos computadores, bem como sua utilidade enquanto ferramentas gráficas. Agora, examinaremos mais de perto sua atuação e indicaremos outros usos para elas.

As funções que nos interessam no momento são o seno e o cosseno; caso ainda não esteja familiarizado com elas, aconselhamos consultar o artigo citado. Nele, você verá como estas funções estão relacionadas com a posição de um ponto ao redor de um círculo, e como usá-las para calcular as coordenadas de qualquer ponto. O programa da bússola, no artigo da página 334, utiliza o seno e o cosseno para posicionar as marcações de número em seus devidos lugares, ao redor da bússola.

COMO DESENHAR UM RELÓGIO

Os programas apresentados até agora não têm muita utilidade prática. Porém, como veremos detalhadamente nos próximos artigos, as funções trigonométricas oferecem variadas possibilidades de uso. Por enquanto, vamos nos concentrar no desenho de um relógio. Você já deve ter pelo menos uma idéia de como fazê-lo, com base no primeiro artigo da série.

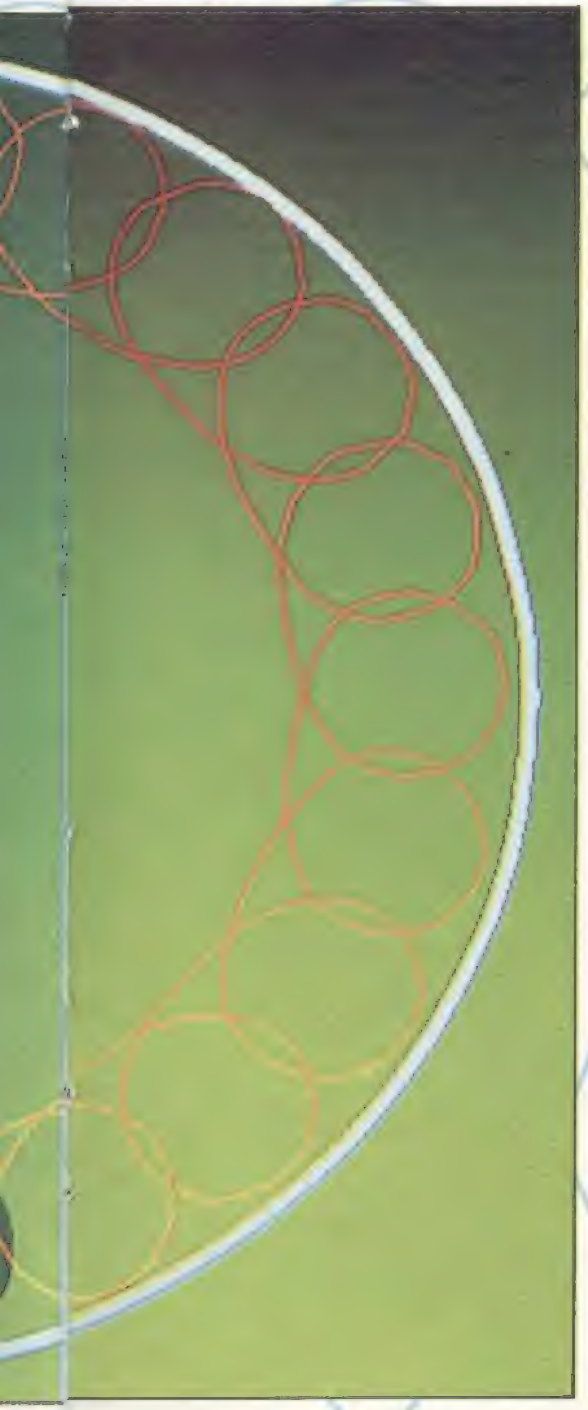
O princípio é o mesmo que orientou o desenho da bússola, com uma diferença: em vez de fornecermos um ângulo, o computador precisará selecionar sozinho os ângulos a serem mostrados, aumentando-os com o tempo.

Os programas a seguir calculam as posições dos ponteiros, usando **SIN** e **COS**, para que mostrem a hora certa.

Embora seja fácil montar um relógio de doze horas, este programa desenha um relógio de 24 horas, com numeração de 1 a 24 em seu mostrador. Dessa maneira, poderemos examinar melhor as rotinas que imprimem os números. Depois de digitar e rodar o programa, veremos o que **SIN** e **COS** fazem exatamente.

No programa da bússola, os números ao redor do aparelho marcavam os vários graus; no relógio, marcam as horas. Os números de 1 a 24 são selecio-

- FAÇA UM RELÓGIO MECÂNICO
- COMO POSICIONAR OS NÚMEROS
- DESENHE OS PONTEIROS
- ACRESCENTE UM ALARME
- COMO ADICIONAR NÚMEROS ROMANOS
- DESENHOS ABSTRATOS
- O USO DE **SIN** E **COS** EM DESENHOS MAIS ELABORADOS



nados por um laço **FOR...NEXT** e impressos na tela em coordenadas determinadas por **SIN** e **COS**.

Parte do programa para o TRS-Color e o MSX parecerão familiares a quem usou a rotina para desenhar caracteres na tela de alta resolução, dada no artigo da página 232. Repetimos esta rotina no programa dado a seguir porque o computador não imprime caracteres na tela em **PMODE 4** ou **SCREEN 2** e **3**, mas com ela podemos colocar os números ao redor do marcador do relógio.

Se você salvou a rotina citada, carregue-a no seu computador para não precisar digitá-la novamente. Depois, acrescente essas linhas ao programa abaixo.

A mesma rotina também serve para colocar números no programa da bússola visto anteriormente; basta adaptá-lo para isso. Quando o marcador é desenhado, o computador automaticamente desenha o ponteiro de segundos do relógio; este funciona como o ponteiro de segundos de um relógio real. O ângulo inicial, no topo do círculo, é de 0 graus, e aumenta gradualmente até chegar nos 360 graus, novamente no topo. A velocidade do ponteiro de segundos é determinada por um comando **PAUSE** ou pelo temporizador no computador.

Os ponteiros do relógio são movimentados por meio de um laço **FOR...NEXT**, no Spectrum. O programa do TRS-Color usa uma variável que é atualizada pelo temporizador cada vez que se roda o programa (este é um laço contínuo). O ponteiro de segundos movimenta-se sempre que a variável acusa que já se passou um segundo.

```
 5 CIRCLE 134,92,70
 6 LET q=-1
10 FOR n=1 TO 24
20 PRINT AT 10-10*COS (n/12*
PI),16+10*SIN (n/12*PI);n
30 NEXT n
40 FOR t=0 TO 20000
50 LET a=t/30*PI
60 LET sx=65*SIN a: LET sy=65
*COS a
70 PLOT 134,92: DRAW  OVER 1;
sx,sy
80 PAUSE 42
 90 PLOT 134,92: DRAW  OVER 1;
 sx,sy
100 NEXT t
```

```
10 PMODE 4,1
20 DIM LE$(26)
30 PCLS
40 FOR K=0 TO 26:READ LE$(K):NE
XT
50 FOR K=0 TO 9:READ NU$(K):NEX
T
60 DATA BR2,ND4R3D2NL3ND2BE2,ND
4R3DGNL2FDNL3BU4BR2,NR3D4R3BU4B
R2,ND4R2FD2GL2BE4BR,NR3D2NR2D2R
3BU4BR2
70 DATA NR3D2NR2D2BE4BR,NR3D4R3
U2LBE2BR,D4BR3U2NL3U2BR2,ND4BR2
,BD4REU3L2R3BR2,D2ND2NF2E2BR2
80 DATA D4R3BU4BR2,ND4FREND4BR2
,ND4F3DU4BR2,NR3D4R3U4BR2,ND4R3
D2NL3BE2,NR3D4R3NHU4BR2
90 DATA ND4R3D2L2F2BU4BR2,BD4R3
U2L3U2R3BR2,RND4RBR2,D4R2U4BR2,
D3FEU3BR2,D4EFU4BR2
100 DATA DF2DBL2UE2UBR2,DFND2EU
BR2,R3G3DR3BU4BR2
110 DATA NR2D4R2U4BR2,BDEND4BR2
,R2D2L2D2R2BU4BR2,NR2BD2NR2BD2R
2U4BR2,D2R2D2U4BR2,NR2D2R2D2L2B
E4,D4R2U2L2BE2BR2,R2ND4BR2,NR2D
4R2U2NL2U2BR2,NR2D2R2D2U4BR2
200 MX=127:SX=127:MY=95:SY=95
210 SCREEN 1,1
220 CIRCLE (127,95),60,1
230 PI=ATN(1)/15
240 FOR K=5 TO 120 STEP 5
250 LINE(127+55*SIN(K*PI),95-55
*COS(K*PI))-(127+59*SIN(K*PI),9
5-59*COS(K*PI)),PSET
260 A$=MID$(STR$(K/5),2):DRAW"B
M"+STR$(INT(123+68*SIN(K*PI)))+
","+STR$(INT(92-68*COS(K*PI)))+
"C5S6":GOSUB 1000
270 NEXT K
280 IF TIMER<50 THEN 280
290 TIMER=0:T=T+1
300 IF T=86400 THEN T=0
310 H=T/3600
320 M=T/60-INT(H)*60
330 S=T-INT(M)*60-INT(H)*3600
340 LX=SX:LY=SY
350 SX=127+45*SIN(S*PI*2):SY=95
-45*COS(S*PI*2)
360 LINE(127,95)-(SX,SY),PSET
370 LINE(127,95)-(LX,LY),PRESET
380 LINE(127,95)-(MX,MY),PRESET
390 MX=127+30*SIN(M*PI*2):MY=95
-30*COS(M*PI*2)
400 LINE(127,95)-(MX,MY),PSET
```

```
410 LINE(127,95)-(HX,HY),PRESET
420 HX=127+20*SIN(H*PI*5):HY=95
-20*COS(H*PI*5)
430 LINE(127,95)-(HX,HY),PSET
440 GOTO 280
1000 FOR M=1 TO LEN(A$)
1010 B$=MID$(A$,M,1)
1020 IF B$>="0" AND B$<="9" THE
N DRAW NU$(VAL(B$)):GOTO 1050
1030 IF B$="" THEN N=0 ELSE N=A
SC(B$)-64
1040 DRAW LE$(N)
1050 NEXT
1060 RETURN
```

```
20 DIM LE$(26)
30 CLS
40 FOR K=0 TO 26:READ LE$(K):NE
XT
50 FOR K=0 TO 9:READ NU$(K):NEX
T
60 DATA BR2,ND4R3D2NL3ND2BE2,ND
4R3DGNL2FDNL3BU4BR2,NR3D4R3BU4B
R2,ND4R2FD2GL2BE4BR,NR3D2NR2D2R
3BU4BR2
70 DATA NR3D2NR2D2BE4BR,NR3D4R3
U2LBE2BR,D4BR3U2NL3U2BR2,ND4BR2
,BD4REU3L2R3BR2,D2ND2NF2E2BR2
80 DATA D4R3BU4BR2,ND4FREND4BR2
,ND4F3DU4BR2,NR3D4R3U4BR2,ND4R3
D2NL3BE2,NR3D4R3NHU4BR2
90 DATA ND4R3D2L2F2BU4BR2,BD4R3
U2L3U2R3BR2,RND4RBR2,D4R2U4BR2,
```

```
D3FEU3BR2,D4EFU4BR2
100 DATA DF2DBL2UE2UBR2,DFND2EU
BR2,R3G3DR3BU4BR2
110 DATA NR2D4R2U4BR2,BDEND4BR2
,R2D2L2D2R2BU4BR2,NR2BD2NR2BD2R
2U4BR2,D2R2D2U4BR2,NR2D2R2D2L2B
E4,D4R2U2L2BE2BR2,R2ND4BR2,NR2D
4R2U2NL2U2BR2,NR2D2R2D2U4BR2
200 MX=127:SX=127:MY=95:SY=95
210 SCREEN2
220 CIRCLE(127,95),60,1,,,1
230 PI=ATN(1)/15
240 FOR K=5 TO 120 STEP5
250 LINE(127+55*SIN(K*PI),95-55
*COS(K*PI))-(127+59*SIN(K*PI),9
5-59*COS(K*PI)),1
260 A$=MID$(STR$(K/5),2):DRAW"B
M"+STR$(INT(123+68*SIN(K*PI)))+
","+STR$(INT(92-68*COS(K*PI)))+
"C1S6":GOSUB 1000
270 NEXT K
280 IF TIME<50 THEN 280
290 TIME=0:T=T+1
300 IF T=86400! THEN T=0
310 H=T/3600
320 M=T/60-INT(H)*60
330 S=T-INT(M)*60-INT(H)*3600
340 LX=SX:LY=SY
350 SX=127+45*SIN(S*PI*2):SY=95
-45*COS(S*PI*2)
360 LINE(127,95)-(SX,SY),1
370 LINE(127,95)-(LX,LY),4
380 LINE(127,95)-(MX,MY),4
390 MX=127+30*SIN(M*PI*2):MY=95
-30*COS(M*PI*2)
400 LINE(127,95)-(MX,MY),1
410 LINE(127,95)-(HX,HY),4
420 HX=127+20*SIN(H*PI*5):HY=95
-20*COS(H*PI*5)
430 LINE(127,95)-(HX,HY),1
440 GOTO 280
1000 FOR M=1 TO LEN(A$)
1010 B$=MID$(A$,M,1)
1020 IFB$>="0" AND B$<="9" THEN
 DRAW NU$(VAL(B$)):GOTO 1050
1030 IFB$="" THEN N=0 ELSE N=AS
C(B$)-64
1040 DRAW LE$(N)
1050 NEXT
1060 RETURN
```

O ZX-81 não possui o comando **DRAW**; assim, para fazer o relógio, precisaríamos desenhar os ponteiros ponto por ponto, via comando **PLOT**. Além disso, sua tela gráfica não é de alta resolução, como nos outros computadores. Por ambos os motivos, uma versão para ele não ficaria muito boa; portanto, não a fizemos.

O capítulo 19 do manual do ZX-81 contém um programa de relógio que poderíamos tomar como exemplo, mas este relógio não tem ponteiros. Ao contrário dos que aqui apresentamos, é formado apenas por números que marcam a hora, sem o círculo como corpo.

O Apple também não possui o comando **DRAW**; por isso, uma rotina para desenhar letras e números em tela de alta resolução ficaria extremamente complicada.

Cada versão emprega uma rotina parecida para imprimir os números em suas posições corretas ao redor do marcador do relógio. Esta rotina mostra claramente como **SIN** e **COS** são usados para controlar as operações de impressão na tela. Ela se resume mais ou menos ao seguinte:

```
10 FOR n=1 TO 24
20 PRINT AT(coordenada x do
   centro da tela + 10*SIN(2*
   PI/n) , coordenada y do cen-
   tro da tela - 10*COS(2*PI/
   n));n
30 NEXT n
```

A maneira de manipular esta rotina varia de computador para computador, o que a faz parecer mais complicada do que realmente é. Determinada máquina pode, por exemplo, ajustar uma variável extra para o ângulo, em vez de usar (2*PI/n) no **PRINT AT**.

A rotina do Spectrum segue a mesma fórmula das outras, mas parece diferente, porque utiliza um método distinto para o **PRINT AT** numa posição da tela: em vez do primeiro número depois do comando ser a coordenada x, é a coordenada y. Além disso, o valor 0 para a coordenada y não corresponde à parte inferior da tela, mas ao topo desta.

Tente isto, no Spectrum:

```
PRINT AT 0,0;"a"
```

O "a" deve aparecer no canto superior esquerdo da tela.

Devido a esta peculiaridade do **PRINT AT** do Spectrum, nele o **COS** é a primeira função da linha, já que determina a posição vertical do ponto ao redor do círculo.

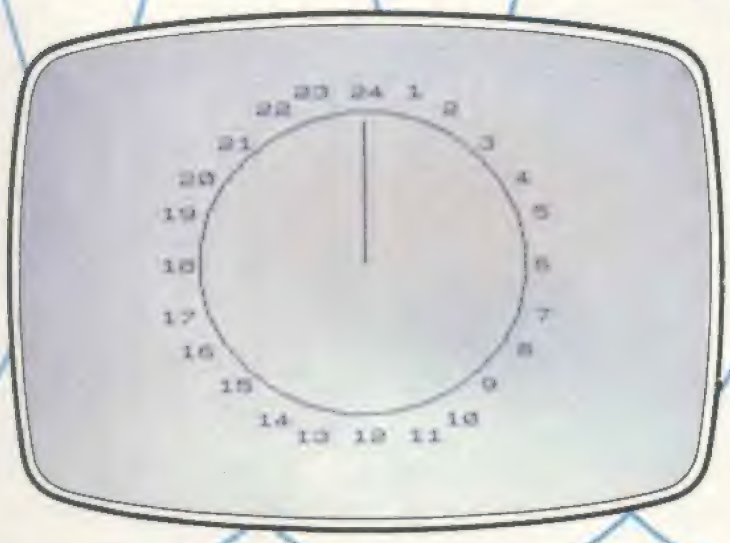
Se compararmos o exemplo acima com os programas, veremos que o laço **FOR...NEXT** que controla o comando **PRINT** é diferente para cada versão. Na verdade, qualquer laço funcionaria em qualquer computador; as diferenças são mantidas por conveniência, conforme o sistema operacional de cada um.

O objetivo da rotina é imprimir números de 1 a 24 ao redor do relógio, marcando as horas. O laço para o programa do Spectrum é:

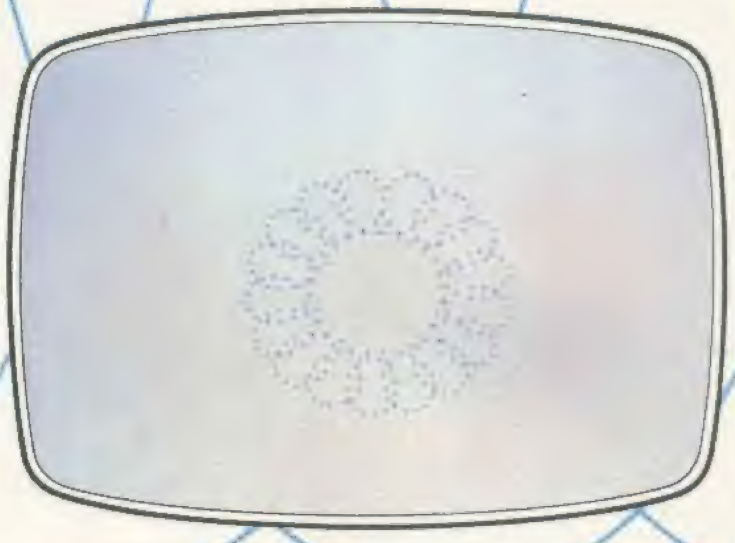
```
FOR n=1 TO 24
NEXT n
```

A rotina calcula a posição para o **PRINT AT** cada vez que passa pelo laço, dividindo o círculo em 24 segmentos.

O TRS-COLOR e o MSX funcionam como o Spectrum mas, como usam uma rotina especial para desenhar letras na



Spectrum: um relógio de 24 horas...



...e espirais em movimento.

tela gráfica, seus programas são um pouco diferentes. Na verdade, eles não imprimem um número relacionado com o **STEP** que controla suas posições angulares. O que imprimem é acessado — ou chamado — pelo comando **STR$**, na linha que controla a impressão. **STR$** simplesmente consulta a rotina que desenha letras; não está relacionado com a posição onde os números são desenhados, a qual é determinada pela variável **K**.

Como o círculo tem 360 graus, para se imprimir 24 números ao redor do relógio, em intervalos iguais, eles devem ser posicionados a cada 360/24 (15) graus. Do mesmo modo, como há 2*PI radianos num círculo completo, os números estão posicionados a cada 2*PI/24, ou PI/12 radianos.

Observe que em todos os laços o primeiro número é o primeiro salto, e não o 0 (1 no Spectrum, 5 no TRS-Color e MSX). Por isso, o primeiro número impresso — ou desenhado, como é o caso do TRS-Color e do MSX — estará em 1 hora e não em 24 horas.

O restante do programa usa **SIN** e **COS** para calcular as posições dos ponteiros, e uma variável ou o temporizador interno do computador para acompanhar o tempo, garantindo, assim, que o ponteiro de segundos se movimente a cada segundo.

COMO ACESSAR OS DADOS

O método para dizer ao computador o que ele deve imprimir não precisa ser aquele utilizado no exemplo do relógio. Poderíamos armazenar os números em **DATA** e depois ler (**READ**) estes números, um de cada vez, à medida que o computador passasse pelo laço **FOR...NEXT**.

Pode parecer um desperdício de memória mas, na verdade, este método nos permite fazer mudanças interessantes. Por exemplo: existem relógios com algarismos romanos no lugar dos números convencionais. Usando uma variável de cadeia, e armazenando em **DATA** as letras, e não os números, poderemos sofisticar ainda mais nosso relógio.

Talvez seja preciso mudar o tamanho e/ou a posição do círculo para que os algarismos romanos caibam na tela (o número 8, por exemplo, é VIII, o que toma bastante espaço!).

Também poderíamos fazer um relógio de 24 horas em algarismos romanos. E é fácil, usando este método, mudar um relógio de 12 para um de 24 horas.

Adicionar um alarme ao programa também não será difícil. Para isso, precisaríamos calcular, usando **SIN** e **COS**, a posição exata em que o ponteiro vai estar na hora ajustada para o disparo do alarme. Uma nova linha com um **IF...THEN** avisaria ao computador que a variável que ajusta a posição do ponteiro é igual à calculada para o alarme e, então, o computador emitiria algum som.

Usando os mesmos princípios, seria possível não só fazer um relógio de 12 horas ou de um dia, mas de uma semana, um mês ou até mesmo um ano. O único limite é a quantidade de informação que podemos colocar na tela.

DESENHOS ABSTRATOS

Até agora, empregamos **SIN** e **COS** para coisas úteis como relógios, bússolas e gráficos. Mas talvez seu uso mais

Espiral no Spectrum.

Flor no MSX.

interessante esteja na elaboração de gráficos abstratos, obtidos com a manipulação adequada das duas funções.

Desenhando séries de círculos e elipses, já produzimos algumas figuras interessantes. Se, por exemplo, movêssemos o centro do círculo ou, então, aumentássemos seu raio a cada volta, o resultado seria ainda melhor. Essas mudanças são muito fáceis de fazer: precisaremos apenas acrescentar algumas linhas às rotinas conhecidas.

COMO FAZER UMA ESPIRAL

No artigo anterior, apresentamos uma rotina que desenha um círculo ponto a ponto. Com algumas mudanças — por exemplo, o aumento gradativo do raio —, poderemos obter uma espiral, em vez de um círculo. As versões a seguir fazem exatamente isto.

```
10 LET z=0
50 FOR n=0 TO 8*PI STEP PI/10
60 PLOT 128+(5+z)*SIN n,88+(5
+z)*COS n
70 LET z=z+1
80 NEXT n
```

```
10 PMODE 4,1
20 PCLS
30 SCREEN 1,1
40 PI=4*ATN(1)
50 FOR N=0 TO 10*PI STEP PI/10
60 PSET(127+(5+Z)*SIN(N),95+(5+
Z)*COS(N),5)
70 Z=Z+1
80 NEXT N
90 GOTO 90
```

```
10  HGR2
20  HCOLOR= 3
30 PI = 4 *  ATN (1)
40  FOR N = 0 TO 8 * PI STEP PI
 / 10
50  HPLOT 140 + (5 + Z) *  SIN
(N),80 + (5 + Z) *  COS (N)
60 Z = Z + 1
70  NEXT N
80  GOTO 80
```

```
10 SCREEN 2
20 COLOR 1,15
30 CLS
```

```
40 PI=4*ATN(1)
50 FOR N=0 TO 10*PI STEP PI/10
60 PSET(128+(5+Z)*SIN(N),96+(5+
Z)*COS(N)),1
70 Z=Z+1
80 NEXT N
90 GOTO 90
```

A diferença entre este programa e o que desenha um círculo é a variável **Z**. Esta começa em 0, aumentando cada vez que o computador passa pelo laço. A mudança que ela traz está nos cálculos da linha 60 no Sinclair, TRS-Color e MSX e da linha 50 no Apple.

O ponto desenhado é controlado, como no círculo, pelo **SIN** e **COS** da variável de controle no laço. O aumento do número que multiplica o **SIN** e **COS**, quando o computador passa pelo laço, faz com que os pontos sejam desenhados cada vez mais longe do centro. A variável **Z** é responsável pelos consecutivos aumentos, que resultam na espiral crescendo para fora.

Como no programa do círculo, podemos obter resultados diferentes se mudarmos o salto (**STEP**) — linha 40 no Apple e 50 nos outros — ou a dimensão do aumento de **Z**. Alguns **STEP** interessantes são 2, 5,PI e 2*PI. Experimente-os e descubra porque os resultados são diferentes (lembre-se de que os computadores trabalham com radianos, e que um círculo tem 2*PI radianos).

Podemos fazer espirais elípticas do mesmo modo que fizemos elipses na primeira versão deste programa: basta mudar o número dentro dos parênteses na linha 60 (linha 50 no Apple).

SOFISTIQUE OS DESENHOS

Existe um brinquedo infantil que utiliza rodas de plástico para desenhar. A criança coloca uma roda menor dentro de uma maior, e a ponta de uma caneta num dos furos da roda menor; movendo a roda menor pelo lado interno da maior, a caneta gira. O desenho obtido é muito interessante.

O centro da roda menor está sempre se movendo quando a criança está desenhando, o que resulta numa série de espirais contínuas. O computador consegue um efeito semelhante, desenhando círculos cujos centros estão em constante movimento.

Os programas que se seguem fazem isso, sendo que, para o Spectrum, a escolha da cor é aleatória.

```
10 CLS : LET x=0: LET p=0:
LET q=0
20 LET z=INT (RND*7): INK z:
LET a=INT (RND*50)
30 LET b=INT (RND*50)
40 FOR n=0 TO 200*PI STEP b/
100
50 IF INKEY$<>"" THEN  GOTO
10
60 LET p=128+(a-b)*SIN n: LET
q=88+(a-b)*COS n
70 PLOT p+b*SIN x,q+b*COS x
80 LET x=x-a/1074
90 NEXT n
110 GOTO 20
```

```
10 PMODE 4,1:PCLS:SCREEN 1,1
20 PI=4*ATN(1)
30 A=RND(50)-1:B=RND(50)-1
40 FOR N=0 TO 200*PI STEP B/100
50 IF INKEY$<>"" THEN 10
60 P=128+(A-B)*SIN(N):Q=95+(A-B
)*COS(N)
70 PSET (P+B*SIN(X),Q+B*COS(X),
5)
80 X=X-A/1074
90 NEXT
100 GOTO 30
```

```
10  HGR2 : HCOLOR= 3
20 PI = 4 *  ATN (1)
30 A =  INT ( RND (1) * 100)
40 B =  INT ( RND (1) * 40)
50  FOR N = 0 TO 200 * PI STEP
B / 100
60 W =  PEEK ( - 16384)
70  POKE  - 16368,0
80  IF W > 127 THEN 10
90 P = 140 + B *  SIN (N):Q = 8
0 + B *  COS (N)
100  HPLOT P + B *  SIN (X),Q +
 B *  COS (X)
110 X = X - A / 1000
120  NEXT N
130  GOTO 30
```

```
10 SCREEN2:COLOR1,15:CLS
20 PI=4*ATN(1)
30 A=INT(RND(1)*200)+20
40 B=INT(RND(1)*40)+10
50 FOR N=0 TO 200*PI STEP B/100
60 IF INKEY$<>"" THEN 10
70 P=128+B*SIN(N):Q=96+B*COS(N)
80 PSET(P+B*SIN(X),Q+B*COS(X)),
1
90 X=X-A/1000
100 NEXT N
110 GOTO 30
```

Nota-se facilmente que o laço **FOR...NEXT** é uma rotina criadora de círculos: os laços para cada computador são

```
FOR N=0 TO 200*PI
```

Como nos outros programas deste artigo que usam **SIN** e **COS**, essas duas funções também estão inclusas no laço. Elas calculam a posição do próximo ponto a ser desenhado, embora, é claro, os cálculos sejam um pouco mais complicados do que um simples "**SIN N**" ou "**COS N**".

Os demais cálculos nesta linha servem para incrementar o resultado da expressão do **SIN** ou **COS**, para que ele possa representar uma posição na tela. Por essa razão, o centro da tela gráfica de cada computador também faz parte dos cálculos. Se olharmos com atenção, encontraremos o centro da coordenada "X" numa metade da linha e o centro da coordenada "Y" na outra.

O cálculo envolve ainda duas variáveis; como elas recebem valores aleatórios no início de cada programa, este sempre executa desenhos diferentes ao ser rodado. Tente mudar o valor dessas variáveis e observe a diferença.

ALTERAÇÕES

Podemos fazer outro tipo de alteração: mudar o **STEP** do laço **FOR...NEXT**. O resultado será o aumento ou a diminuição da densidade dos pontos, conforme o **STEP** seja aumentado ou diminuído. Para a obtenção do desenho, o centro do círculo deve mudar cada vez que um ponto é desenhado. Para isso, colocamos o próprio centro do círculo num outro círculo, que gira a cada passo do laço. O efeito é produzido pelo segundo grupo de cálculos do **SIN** e **COS**, baseado em x. Essa variável decresce levemente cada vez que o computador passa pelo laço.

Tente alterar o tanto que a variável (x) diminui e veja o que acontece com os desenhos. Estes seis valores trazem resultados interessantes: A/10, A/−10, A/−50, A/0.5, A/0.1, A/0.001.

Todas as versões permitem que reiniciemos o programa a qualquer hora, bastando apertar uma tecla. O Spectrum, o TRS-Color e o MSX usam o **INKEY$** para verificar se alguma tecla foi pressionada, enquanto o Apple busca essa informação na memória (linhas 60 a 80, já comentadas em artigos anteriores sobre **PEEK** e **POKE**).

Examinaremos ainda diferentes usos para funções trigonométricas e angulares. No próximo artigo desta série, trataremos de quadrados, cubos, raízes quadradas e outras ferramentas poderosas para controlar o comportamento das variáveis. Você terá, assim, novas oportunidades de dar às funções empregos que, à primeira vista, não parecem estar relacionados com a matemática.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC

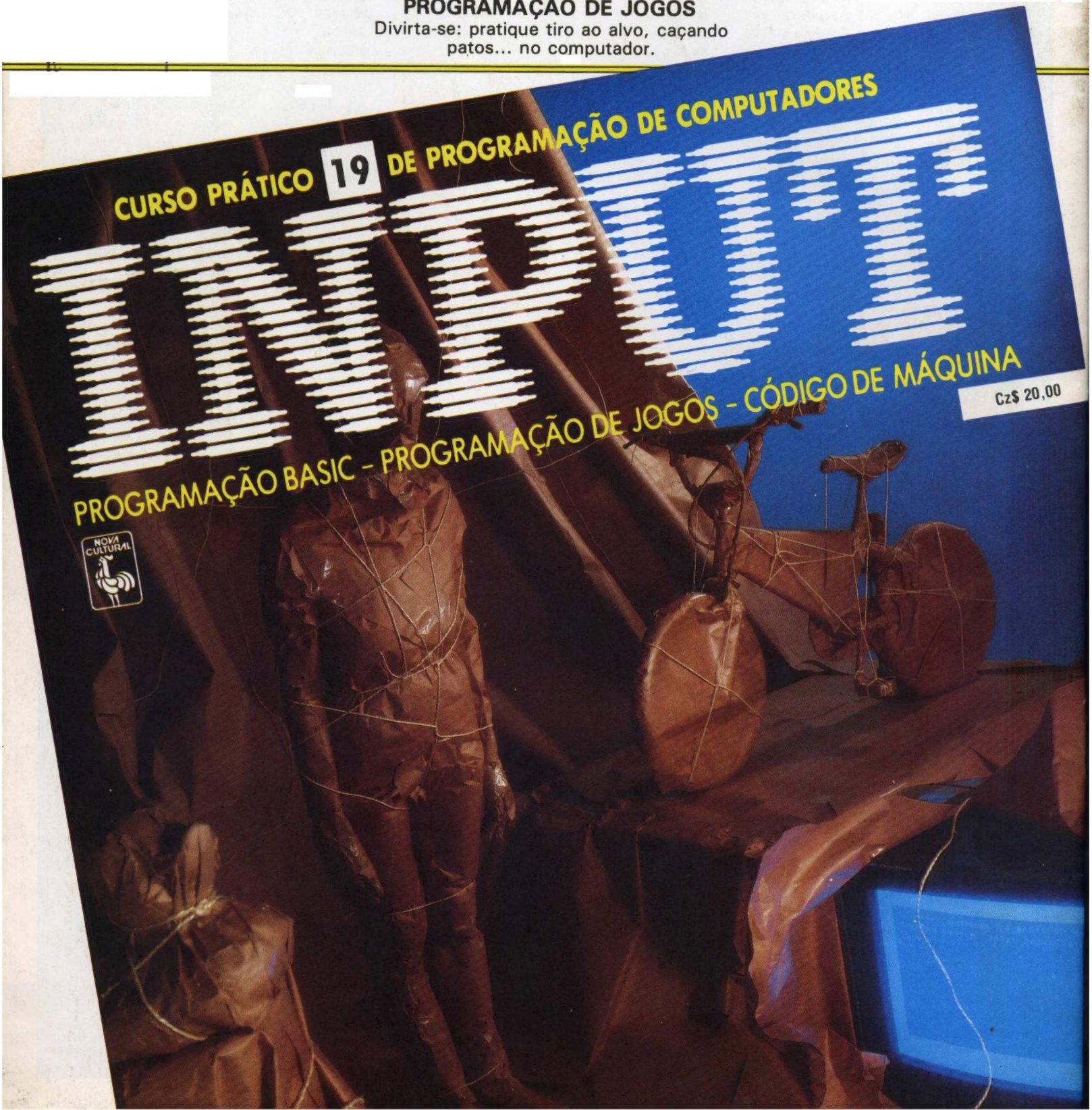
Os códigos de controle são muito úteis para uma série de truques de programação. Veja como usá-los nos micros MSX.

APLICAÇÕES

Você acha difícil converter metro em polegadas ou galão em litros? Nosso programa conversor resolve esse problema.

PROGRAMAÇÃO BASIC

O código ASCII foi concebido para possibilitar a transferência de dados de um computador para outro. Aprenda a utilizá-lo.

PROGRAMAÇÃO DE JOGOS

Divirta-se: pratique tiro ao alvo, caçando patos... no computador.